Amatérské RADIO

NOSITEL VYZNAMENÁNÍ ZA BRANNOU VÝCHOVU I. A II. STUPNĚ

ŘADA B PRO KONSTRUKTÉRY

ČASOPIS PRO ELEKTRONIKU A AMATÉRSKÉ VYSÍLÁNÍ

ROČNÍK XXXII/1983 ● ● ČÍSLO 2

V TOMTO SEŠITĚ

AMATÉRSKÉ RADIO ŘADA B

Vydává ÚV Svazarmu ve vydavatelství NAŠE VOJSKO, Vladislavova 26, 133 66 Praha 1, tel. 26 06 51–7. Šéfredaktor ing. Jan Klabal, redaktor Luboš Kalousek, OK1FAC. Redakční rada: RNDr. V. Brunnhofer, V. Brzák, K. Donát, V. Gazda, A. Glanc, I. Harminc, M. Háša, Z. Hradiský, P. Horák, J. Hudec, ing. J. T. Hyan, ing. J. Jaroš, doc. ing. dr. M. Joachim, ing. F. Králík, RNDr. L. Kryška, J. Kroupa, ing. E. Môcik, V. Němec, RNDr. Ľ. Ondriš, CSc., ing. F. Smolík, ing. E. Smutný, ing. V. Teska, doc. ing. J. Vackář, laureát st. ceny KG, J. Vorlíček.

Redakce Jungmannova 24, 113 66 Praha 1, tel. 26 06 51–7, šéfredaktor linka 354, redaktor linka 353, sekretářka linka 355. Ročně vyjde 6 čísel. Cena výtisku 5 Kčs, pololetní předplatné 15 Kčs. Rozšiřuje PNS, v jednotkách ozbrojených sil vydavatelství NAŠE VOJSKO, administrace Vladislavova 26, Praha 1. Objednávky přijímá každá pošta i doručovatel. Objednávky do zahraničí vyřizuje PNS, ústřední expedice a dovoz tisku, závod 01, Kafkova 9, 160 00 Praha 6. Tiskne NAŠE VOJSKO, n. p., závod 08, 162 00 Praha 6, Vlastina 710.

Za původnost a správnost příspěvku odpovídá autor. Návštěvy v redakci a telefonické dotazy pouze po 14. hodině.

Číslo indexu 46 044.

Toto číslo má vyjít podle plánu 16. 3. 1983.

© Vydavatelství NAŠE VOJSKO

VSTŘÍC VII. SJEZDU SVAZARMU

V souladu se stanovami branné organizace bude v tomto roce 1983 svolán VII. celostátní sjezd Svazarmu. Sjezdu bude předcházet roční období, v němž budou připraveny a provedeny výroční členské schůze základních organizací, okresní konference, krajské konference a sjezdy obou republikových organizací. Kampaň bude probíhat v závěru pětiletého období, které uplyne od VI. celostátního sjezdu Svazarmu, a ve třetím roce 7. pětiletky. Svazarmovci musí udělat vše, aby se mohutný rozvoj aktivity a iniciativy na závodech, stavbách, v zemědělství i v dopravě za splnění úkolů XVI. sjezdu strany plně projevil také v jejich aktivitě na počest VII. sjezdu Svazarmu.

Základní orientací jednotného přístupu k obsahové přípravě a jednání výročních členských schůzí, konferencí a sjezdů budou tvořit úkoly, jež vyplynuly ze zasedání plén ústředního výboru Svazarmu po VI. sjezdu a z rozpracování závěrů XVI. sjezdu KSČ do podmínek branné organizace. Celá kampaň se *musí stát školou ideově politické výchovy*, v níž nebudou řešeny pouze organizační stránky připravovaných jednání, ale také politickovýchovná práce v celém hnutí.

Bude třeba udělat vše, aby se schůze, konference a sjezdy staly náročným a kritickým posouzením realizace úkolů plynoucích z rezoluce VI. sjezdu a jednotlivých zasedání ústředního výboru Svazarmu. V průběhu kampaně bude proto provedena všestranná analýza práce všech řídicích orgánů, organizací a klubů. Stejná péče bude věnována práci odborně metodických orgánů s hlavní pozorností k plnění úkolů plynoucích z poslání Svazarmu ve společnosti.

Vysoká ideová a organizační úroveň celé kampaně bude do značné míry záviset na tom, jak budou jednotlivé orgány zabezpečovat její přípravu, jak se jim podaří tyto mimořádně důležité úkoly na jednotlivých stupních plánovat a zodpovědně připravit *dobrovolné funkcionáře, pracovníky aparátu* a členy na provedení kampaně.

Velký význam bude mít nejen kvalita a otevřenost zpracovaných analytických materiálů, ale i atmosféra každého jednání. Proto je zapotřebí zajistit, aby diskuse byla obrazem soudružské spolupráce, principiálnosti, konstruktivní kritiky a sebekritiky i tribunou výměny cenných zkušeností. Všem schůzím, konferencím a sjezdům by mělo být cizí jakékoli okázalé řečnění, slavnostní projevy a samoúčelné tzv. prodávání úspěchů v různých oblastech naší činnosti.

Nejvýznamnějším úkolem bude zhodnotit prohlubování společenské funkce Svazarmu, závislé především na poskytování účinnější pomoci ozbrojeným silám při zajišťování spolehlivé obrany socialistické vlasti. Proto bude třeba na všech výročních jednáních hledat především optimální cesty a metody k rozšiřování branného vlivu organizace na široké *vrstvy občanů, zvláště mládeže, způsoby dalšího* prohloubení společného úsilí a spolupráce se státními orgány, organizacemi Národní fronty, zejména se Socialistickým svazem mládeže, Revolučním odborovým hnutím, ČSTV, ČSVTS, školami a závody. Přitom je třeba mít na zřeteli zejména vytváření a prohlubování odpovídajících forem spolupráce orgánů a základních organizací Svazarmu s útvary Československé lidové armády, vojsk ministerstva vnitra a Lidovými milicemi, směřujících ke kvalitnímu splnění společných úkolů ve prospěch obrany země.

V zájmu účinnějšího a cílevědomějšího utváření třídního, vlasteneckého a internacionálního uvědomění členů třeba zhodnotit *dosavadní výsledky*, možnosti a zkušenosti a zvolit nejúčinnější metody politickovýchovné práce do budoucna. Výroční jednání musí svoji pozornost zaměřit také na agitačně propagační práci, na zvýšení úrovně ideově politické *přípravy funkcionářského aktivu* i členů s důrazem na všestranné pochopení závěrů XVI. sjezdu strany a jejich rozpracování orgány ústředního výboru Svazarmu do našich podmínek.

Významnou oblastí, kterou projednávají všechny orgány a základní organizace s maximální pozorností, je daleko masovější rozvoj zájmových branných činností. Celá kampaň schůzí a konferencí *může přispět rozhodujícím způsobem k tomu*, abychom úkol masovosti a vyšší účinnosti v této oblasti splnili.

Jednotlivé odbornosti zájmové branné činnosti musí postupně naplnit požadavky výraznější orientace na vyšší úroveň polytechnické výchovy ve Svazarmu. Přednost musí základní organizace dát rozvoji těch činností, které nekladou zvláštní nároky na náročnou techniku a zařízení a umožňují široké branné využití mládeže i ostatních občanů. Všechna jednání schůzí a konferencí musí podněcovat a rozvíjet zájem veřejnosti a zvláště mládeže o progresívní techniku, zejména o masovější rozvoj disciplín elektroniky. Přitom nutno ukázat konkrétní cesty a prostředky k prosazení ideovosti a politočnosti do činnosti jednotlivých zájmových odborností.

Ústřední výbor Svazarmu zdůraznil v usnesení 10. pléna, že rozvíjení polytechnické výchovy ještě plně neodpovídá soudobým požadavkům vědeckotechnického rozvoje. *Některé orgány a organizace* nenaplňují obsah koncepcí jednotlivých odborností komplexně. V období předsjezdové kampaně proto ÚV Svazarmu zavázal orgány a organizace i jednotlivé odbornosti k zhodnocení dosavadní činnosti podle schválených koncepcí tak, aby byly důsledněji plněny ve všech jejich požadavcích. K zvýšení podílu polytechnické výchovy 10. plénum ve svém usnesení uložilo:

1. Sekretariátu ÚV Svazarmu (kromě jiného) s ohledem na potřeby polytechnické výchovy posoudit vývojové a výrobní programy podniků ÚV Svazarmu a v časopisech ÚV Svazarmu zajistit kvalitní technickou propagandu.

2. *Republikovým ÚV, KV a OV Svazarmu:*

Analyzovat úroveň polytechnické výchovy v působnosti příslušného územního orgánu. Úkoly, vyplývající z této analýzy a zprávy schválené 10. zasedáním ÚV Svazarmu zapracovat do usnesení VČS ZO, konferencí a aktivů odborností a plánů činnosti. Zaměřit se zejména na tyto úkoly:

- posoudit stávající stav úrovně polytechnické propagandy v časopisech republikových organizací Svazarmu a zajistit její rozšíření a zkvalitnění. Organizovat dopisovatelskou činnost a polytechnickou tématiku do časopisu ÚV Svazarmu.
- analyzovat potřeby přípravy kádrů pro rozšiřování a zkvalitňování polytechnické výchovy a zajistit *přípravu cvičitelů a instruktorů*.
- organizovat výstavbu objektů a materiálně technické základny pro polytechnickou výchovu, zejména modelářských dílen, kabinetů elektroniky, leteckých a para učeben a svépomocných motoristických dílen. Výstavby těchto zařízení konkretizovat v ročních plánech.
- pro polytechnickou výchovu využívat objekty jiných organizací a podniků, zejména škol a výrobních závodů, závodních klubů ROH apod. K tomu uzavírat s těmito organizacemi plány.
- prověřit náplň vedlejší hospodářské činnosti ZO a orientovat ji ve větší míře na výrobu potřeb pro polytechnickou výchovu.
- v souladu s dohodou mezi ÚV Svazarmu a MNO rozšiřovat přípravu svazarmovských kádrů u vojenských útvarů. Rozšiřovat spolupráci s těmito útvary při získávání cvičitelů pro polytechnickou výchovu.

3. *Ústředním radám, sekcím a komisím:*

- v souladu s obsahem koncepcí dalšího rozvoje odborností posoudit dosavadní náplň úkolů v polytechnické výchově a přijmout opatření k jejímu zkvalitnění a dalšímu rozšíření v duchu přijaté *zprávy*. Tato opatření projednat na aktivech a konferencích rad a sekcí u všech územních orgánů a zapracovat do usnesení.
- prověřit a doplnit požadavky na výrobu a dodávky materiálů pro polytechnickou výchovu s cílem realizovat ji ve výrobních podnicích Svazarmu, v základních organizacích s vedlejší hospodářskou činností, případně průmyslových podnicích,

družstevních organizacích s aktivním podílem správy podniků Svazarmu.
- rozvíjet zájmovou činnost již od věku 8–10 let tak, aby nejmladší generace byla připravena splnit svůj podíl na rozvoji národního hospodářství i zabezpečení obrany socialistické společnosti. K tomu upřesnit programy výcviku.
- diferencovaně pracovat s mládeží předvojenského věku, tj. od 15–19 let, aby si cílevědomě prohlubovala polytechnické znalosti. K tomu rozpracovat potřebná opatření.

Pro výbory základních organizací a řídicí orgány všech stupňů budou výroční schůze, konference a sjezdy zkouškou zralosti a prověrkou způsobilosti kvalitně zabezpečit plnění stanovených úkolů. Ústřední výbor Svazarmu je přesvědčen o tom, že celý funkcionářský aktiv, pracovníci Svazarmu a všechny řídicí orgány v plné míře pochopí význam a vážnost nastávajícího období a udělají vše pro úspěch a účinnost celé předsjezdové kampaně a zdar VII. celostátního sjezdu Svazarmu.

MIKROPROCESORY A MIKROPOČÍTAČE

III. Mikropočítačový systém JPR-1

Ing. Eduard Smutný

ALFANUMERICKÁ KLÁVESNICE ANK-1

ANK-1 je podle mého názoru klíčem ke stavbě mikropočítače JPR-1. Ať už si bude systém JPR-1 stavět amatér nebo profesionál, bude potřebovat alfanumerickou klávesnici. Klávesnice vyráběné u nás jsou pro mikropočítače příliš drahé a také příliš dobré. Mikropočítač nepotřebuje, aby součástí klávesnice byla složitá elektronika, která převede kód 1 z *N* na normalizovaný kód pro písmena i číslice. Nepotřebuje také, aby elektronika „ošetřila" zákmity kontaktů. Na to všechno stačí přece jeho mikroprocesor. Mikropočítače musíme využít jak jen to jde. Používáme-li mikropočítač jako vývojový systém pro sestavování a ladění programů, je jeho časové využití minimální. Já říkám, že se vlastně většina mini a mikropočítačů vlastně pořád „fláká". Většinu času totiž čeká na dokončení operace přídavných zařízení nebo na příkaz obsluhy, přerušení apod. To, co pak má udělat, stihne za zlomek sekundy a opět na něco čeká.

Jako příklad uvedu řízení krokového motoru. První desky pro připojení krokových motorů obsahovaly čítač, do něhož počítač nahrál počet impulsů a pak dal povel start. Čítače impulsy posílané do krokového motoru počítaly směrem dolů a až se dosáhlo nuly, dala deska počítači hlášení READY. Později již funkci čítače plnil počítač a na desce pro připojení krokových motorů byl jen rozdělovač a monostabilní obvod. Počítač poslal START a po periodě kroku dal monostabilní obvod signál READY. I tak však nebyl počítač využit. Musel nakonec převzít i funkci rozdělovače impulsů a vytvářet tak správný sled fází krokového motoru i funkci časovače. Deska připojení pak vlastně obsahovala pouze čtyři výkonové spínače připojené na čtyři výstupy portů mikropočítače. Dnes se používá tzv. mikropolohování, při němž jsou fáze krokových motorů přes převodníky D/A napájeny „analogovým" proudem. Motor tak může zaujmout nikoli pouze 200 poloh na jednu otáčku, ale třeba 3200, neboť se jeden krok velikostí proudu rozdělí na 16 poloh. U těchto systémů mikropočítač generuje celý tvar analogového průběhu fázového proudu krokového motoru. Je vidět, že využít mikropočítače je skutečně umění, kterému je nutno se učit. Úloha určit, které tlačítko bylo na klávesnici stlačeno, je pro programování základním příkladem maximálního využití možností počítače. Klávesnice ANK-1 je z hlediska hardware příkladem maticového uspořádání klávesnice, které je nejjednodušší a také u mikropočítačů nejrozšířenější. ANK-1 je „ASKI KLÁVESNICE", a proto začnu u významu těchto dvou tak často vyslovovaných slov.

Kód ASCII

Zkratka ASCII znamená „American Standard Code for Information Interchange" a vyslovuje se „aski". Tento kód platí jako USA Standard X3.4 – 1968 a díky tomu, že tato norma byla připravena s ohledem na snadné dekódování skupin znaků i s ohledem na potřeby komunikačních i datových systémů a jednoznačné zobrazení, rozšířila se velmi rychle do všech oblastí výpočetní techniky. U nás platí obdobné kódy ISO-7 a MTA-5.

Kód ASCII je sedmibitový a je ho možné tedy doplnit paritou pro zabezpečení správného přenosu. Obvykle se zobrazuje ve formě tabulky, která má 8 sloupců po 16 řádcích (tab. 1). Z tabulky je zřejmý charakter kódu:

a) první dva sloupce jsou tvořeny znaky, které nemají grafické zobrazení a slouží jako řídicí nebo služební znaky,
b) dalších šest sloupců je tvořeno grafickými znaky,
c) výjimečným znakem je DEL, který je představován sedmi jedničkami a slouží k předěrování chybného znaku na děrné pásce,
d) pro jednoduchá přídavná zařízení je možno použít šestibitový kód (sloupce 2, 3, 4 a 5),
e) kód se jednoduše dekóduje podle jednotlivých sloupců. Dekódování řádků je usnadněno dodržením binárního vyjádření čísel 0 až 9. Bohužel totéž neplatí u písmen A až F, které se u mikropočítačů používají pro hexadecimální vyjádření binárních čísel. Z těchto důvodů používá např. počítač JPR-1 písmena B až G, protože pak stačí udělat „logické nebo" mezi bity 4 a 7 a dostaneme binární vyjádření všech 16 znaků.

Protože se v literatuře i v listingu programů objevují i názvy jednotlivých znaků ASCII, jsou v tab. 2 názvy v angličtině a jejich vyjádření HEX a DEC je v tab. 3.

Tab. 1. Tabulka kódu ASCII

BIT b_7 b_6 b_5					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4	b_3	b_2	b_1	řádek / sloupec	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	–	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	–	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

Tab. 2. Názvy řídicích znaků kódu ASCII

NUL	= All zeros
SOH	= Start of heading
STX	= Start of text
ETX	= End of text
EOT	= End of transmission
ENQ	= Enquiry
ACK	= Acknowledgement
BEL	= Bell or attention signal
BS	= Back space
HT	= Horizontal tabulation
LF	= Line feed
VT	= Vertical tabulation
FF	= Form Feed
CR	= Carriage return
SO	= Shift out
SI	= Shift in
DLE	= Data link escape
DC 1	= Device control 1
DC 2	= Device control 2
DC 3	= Device control 3
DC 4	= Device control 4
NAK	= Negative acknowledgement
SYN	= Synchronous/idle
ETB	= End of transmitted block
CAN	= Cancel (error in data)
EM	= End of medium
SUB	= Start of special sequence
ESC	= Escape
FS	= Information file separator
GS	= Information group separator
RS	= Information record separator
US	= Information unit separator
DEL	= Delete

Řídicí znaky kódu ASCII je možno rozdělit do 4 skupin: 1. řízení komunikace, 2. ovládání formátu (tiskárny, psací stroje), 3. řízení přídavných zařízení, 4. oddělovače informací.

Znaky patřící do první skupiny se používají převážně při přenosu dat. Znak NUL, představovaný samými nulami, slouží k vyplnění volného času nebo k vyplnění prázdného média. Znaky NUL mohou být doplněny nebo vzaty ze sekvence znaků, aniž by se tím změnil význam této sekvence (příklad: doplnění prázdné děrné pásky mezi programy a jejich vypuštění při výpisu pásky). Znak SYN slouží k synchronizaci znaků v synchronní komunikaci. Když přijímací stanoviště rozpozná synchronizační znak, zasynchronizuje se na přijímanou sekvenci a může začít přenos dat. Znak SOH značí začátek záhlaví zprávy. V tomto záhlaví je obvykle uvedena adresa a další údaje o zdroji a směru zprávy. Znak STX ukončuje záhlaví a označuje začátek vlastní zprávy. Zpráva končí znakem ETX – konec textu, nebo, jde-li o blokový přenos, EOB – konec bloku. Je-li ukončen celý přenos, je poslán ještě znak EOT – konec přenosu. Znak ENQ znamená jednak požadavek, aby vzdálené stanoviště vyslalo znaky, které ho umožní identifikovat, a jednak stav připravenosti. Znaky ACK a NAK jsou vysílány přijímacím stanovištěm vysílajícímu jako odezva, buď na ukončený konec bloku dat, nebo na znak ENQ. Odezva může být kladná, ACK, nebo záporná, NAK. Chceme-li do sekvence znaků v kódu ASCII vsunout znaky, které mají jiný význam než je definováno (třeba speciální grafické symboly), musíme vyslat nejprve znak SO (shift-out) a po vyslání speciálních znaků znak SI (shift-in). Pro podobné účely slouží znaky DLE a ESC, které označují, že je změněn význam následujících znaků. Tato sekvence se obvykle ukončuje opět znakem SI. Znak CAN umožňuje označit data, v nichž je chyba, a jež mají být přijímacím stanovištěm ignorována.

Klávesnice

Klávesnice je obvykle řešena jako matice spínacích prvků. Pro mikropočítače jsou nejvhodnější klávesnice vytvořené sítí na sebe kolmých vodičů, které tvoří *n* řádků a *m* sloupců. Je-li stlačeno tlačítko, spojí se kontakt v průsečíku souřadnic, takže napětí přivedené na řádkové vodiče je možno sejmout na sloupcových vodičích. Sekvenčním připojováním řádkových vodičů (třeba na úroveň log. 0) a čtením úrovní na sloupcových vodičích je možno určit, které tlačítko je stlačeno. Klávesnice se vyrábějí v mnoha provedeních a konstruktéři již vyzkoušeli téměř všechny fyzikální principy, aby dosáhli co největší spolehlivosti spínání.

Nejjednodušší je mechanický kontakt. Klávesnice používající tento princip jsou levné a konstrukčně jednoduché. Protože však kontakt je ve styku s okolním prostředím, jsou doba života i spolehlivost těchto klávesnic malé. Mechanický kontakt také dlouho zakmitává, takže je třeba použít buď filtrační logiku nebo zpožďovací rutinu v programu. Tyto klávesnice, používají-li zlacený kontakt, mají dobu života 5 až 10 miliónů operací.

Klávesnice s mechanickým kontaktem tvořeným spínačem z jazýčkového relé jsou velmi rozšířeny zejména u elektronických pokladen a všude tam, kde je požadována nízká cena a odolnost proti okolnímu prostředí. Sepnutí kontaktů je zajištěno malým magnetem, připevněným na pohyblivé části tlačítka. Jelikož je kontakt zataven hermeticky ve skle, není ovlivňován prachem a agresívními prvky ve vzduchu. Zakmitávání kontaktů je krátké díky vysokému rezonančnímu kmitočtu malého jazýčku.

Klávesnice pracující na principu saturace jádra používají miniaturní toroidní transformátorky. Není-li tlačítko stlačeno, je jádro v saturaci, takže přenos transformátorku je minimální. Podobně jako u jazýčkových kontaktů zajišťuje saturaci malý magnet připevněný na pohyblivé části tlačítka. Stlačením tlačítka se magnet posune, jádro už není v saturaci a napětí vysokého kmitočtu se přenese na sekundární vinutí. Tyto klávesnice jsou v zahraničí značně rozšířeny a vynikají spolehlivostí.

Další technologie, založená na magnetismu, používá spínače pracující na principu Hallova efektu. Klávesnice jsou energeticky velmi náročné, protože všechny obvody jsou trvale napájeny. Tato nevýhoda bývá vyvážena spolehlivostí (často až 100 miliónů operací). Uvedený princip se používá u klávesnic vyráběných u nás v ZJS Brno. Klávesnice jsou náročné na stabilitu napájecího napětí, jsou drahé a pro amatéry nedostupné.

„Kapacitní" klávesnice používají podobný princip jako klávesnice se saturovaným jádrem. Vazebním prvkem mezi zdrojem napětí vysokého kmitočtu a snímacím obvodem je však místo transformátorků kondenzátor. Kapacita kondenzátoru se mění stlačením tlačítka, které přiblíží pohyblivou elektrodu k pevné, ta je obvykle vyleptána na desce s plošnými spoji. Klávesnice vyžadují speciální snímací zesilovače nebo alespoň obvody CMOS. Princip kapacitních klávesnic byl v zahraničí velmi oblíben a dnes začíná jeho renezance ve spojení s membránovými klávesnicemi.

Membránové klávesnice, založené zatím převážně na kontaktním principu, si razí cestu na špičku odbytu ve světě. Mohou být vyráběny za neobyčejně nízkou cenu, jsou odolné proti prachu, proti vlhkosti, oleji atd. Jejich hlavní výhodou je, že odstraňují nutnost popisovat každý hmatník jiným symbolem. Grafickým počítačovým systémem je možno zhotovit výrobní podklady pro libovolné rozmístění, velikost a popis tlačítek během jednoho dne. V prostředí, v němž se předpokládají špinavé ruce (u obráběcích strojů, v domácnosti, v lékařství) si tyto klávesnice získaly dominantní postavení. Jediná jejich nevýhoda spočívá v tom, že tlačítka nemají obvyklý zdvih, ale jen asi 0,1 až 0,2 mm. Proto se nepoužívají tam, kde je obsluhují rychlé písařky, které mají ve zvyku mít prsty připraveny na několika klávesách najednou. Jsou-li membránové

Tab. 3. Kód ASCII ve vyjádření HEX a DEC

HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC	ASCII
00	0	NUL	20	32	Space	40	64	@	60	96	\
01	1	SOH	21	33	!	41	65	A	61	97	a
02	2	STX	22	34	"	42	66	B	62	98	b
03	3	ETX	23	35	#	43	67	C	63	99	c
04	4	EOT	24	36	S	44	68	D	64	100	d
05	5	ENQ	25	37	%	45	69	E	65	101	e
06	6	ACK	26	38	&	46	70	F	66	102	f
07	7	BEL	27	39	'	47	71	G	67	103	g
08	8	BS	28	40	(	48	72	H	68	104	h
09	9	HT	29	41	)	49	73	I	69	105	i
0A	10	LF	2A	42	*	4A	74	J	6A	106	j
0B	11	VT	2B	43	+	4B	75	K	6B	107	k
0C	12	FF	2C	44	,	4C	76	L	6C	108	l
0D	13	CR	2D	45	–	4D	77	M	6D	109	m
0E	14	SO	2E	46	.	4E	78	N	6E	110	n
0F	15	SI	2F	47	/	4F	79	O	6F	111	o
10	16	DLE	30	48	0	50	80	P	70	112	p
11	17	DC1	31	49	1	51	81	Q	71	113	q
12	18	DC2	32	50	2	52	82	R	72	114	r
13	19	DC3	33	51	3	53	83	S	73	115	s
14	20	DC4	34	52	4	54	84	T	74	116	t
15	21	NAK	35	53	5	55	85	U	75	117	u
16	22	SYN	36	54	6	56	86	V	76	118	v
17	23	ETB	37	55	7	57	87	W	77	119	w
18	24	CAN	38	56	8	58	88	X	78	120	x
19	25	EM	39	57	9	59	89	Y	79	121	y
1A	26	SUB	3A	58	:	5A	90	Z	7A	122	z
1B	27	ESC	3B	59	;	5B	91	[	7B	123	{
1C	28	FS	3C	60	<	5C	92	\	7C	124	|
1D	29	GS	3D	61	=	SD	93	]	7D	125	}
1E	30	RS	3E	62	>	3E	94	↑	7E	126	~
1F	31	US	3F	63	?	5F	95	←	7F	127	DEL

klávesnice použity v zařízení, obvykle je stlačení „tlačítka" doprovázeno zvukovým signálem z mikropočítače. V poslední době se však začínají objevovat membránové klávesnice s obvyklým zdvihem (a to buď na kapacitním nebo kontaktním principu), u nichž se používají běžná tlačítka a membrána zajišťuje jen odolnost proti prostředí a snadnou výrobu kontaktního pole.

Klávesnice využívající vodivých elastomerů si hledají cestu vpřed již několik let, ale bez větších úspěchů. Mají větší zdvih než membránové, jsou však také dražší a vyžadují speciální nástroje a materiál.

Pro amatéry je u nás nejdostupnější membránová klávesnice, všechny ostatní vyžadují popsat hmatníky tlačítek jednotlivými znaky. Pro membránovou klávesnici stačí negativ filmu, vybarvený zespodu modelářskými barvami. Protože se však u nás na membránové klávesnice zapomnělo i v průmyslu, není k dispozici fólie ani jiný způsob, jak membránu vytvořit. Jak vlastně vypadá membránová klávesnice? Základní nosnou deskou je běžný kuprextit s fólií Cu. Měl by být oboustranně plátovaný s prokovenými děrami, aby bylo možné propojit spínače do matice. Na jeho horní straně jsou vytvořeny kontaktní meandry různých tvarů. Nejjednodušší jsou dva hřebínky zasunuté do sebe, hřebínky (meandry) se však nesmí dotýkat; spojí se, až když na ně shora přilehne vodivá membrána. Membrána by mohla být vytvořena vodivou elastickou fólií bez jakéhokoli členění. Membrána nemusí být s ničím spojena, zajišťuje pouze zkrat pevných kontaktních meandrů. K zajištění elastičnosti vodivého materiálu fólie se však i zkratovací plošky dělají buď jako čárky, soustředné kruhy nebo „pavoučci". Důležitá je technologie nanášení vodivé vrstvy na fólii membrány. Původní způsob používal měď plátovanou na pružný materiál (Mylar), která se odleptávala jako při technologii ohebných plošných spojů. Dnes se již tato technologie v zahraničí nepoužívá. Byly vyvinuty pasty (obdobné těm, které se používají u hybridních obvodů), které se na elastickou fólii nanesou sítotiskem a vypálí se. Pasty jsou buď stříbrné nebo uhlíkové jako u potenciometrů, neboť současné obvody MOS a CMOS nejsou příliš náročné na velikost odporu sepnutého kontaktu. Pasty musí být po vypálení pružné.

Teď již víme, že principem membránové klávesnice je deska s plošnými spoji, která tvoří mechanický základ klávesnice a membrána, která má zkratovací plošky. Mezi tyto dva prvky se uloží oboustranně lepicí vložka (papír, plastická hmota) s otvory nad kontaktními meandry a vše se slepí dohromady. Na membránu se přilepí další fólie, která má zespodu sítotiskem vytvořený popis jednotlivých tlačítek v libovolných barvách. Na obr. 1 je řez membránovou klávesnicí a v tab. 4 typické parametry membránových klávesnic. Vývody z klávesnice jsou vedeny buď vodiči přímo ze základní desky s plošnými spoji nebo přímým konektorem přes zlacené plošky této desky. Některé klávesnice používají i pro základní kontaktní systém fólii, a pak je vývod přímo fólií, obdobně jako u ohebných desek s plošnými spoji. Navrhujeme-li zařízení, které bude mít membránovou klávesnici a bude se vyrábět ve velkých sériích, můžeme jako základní desku klávesnice použít desku s plošnými spoji, která nese ostatní součástky. Získáme tak kompaktní celek, který nemá spoje a je laciný a spolehlivý (osobní počítač ZX-80).

Pro amatéry je pružná plastická fólie s nanesenou měděnou fólií nedostupná. Proto jsme vymysleli klávesnici, která používá místo membrány hliníkovou fólii (Alobal). Nebylo by na závadu, kdyby vaše pokusy s membránovými klávesnicemi přinesly nové nápady a návody, pomohlo by to i naší elektronice.

Stavba amatérské membránové klávesnice

Na obr. 2 je znázorněno, z jakých dílů se membránová klávesnice sestaví. Díl *1* je hmatník, který je otištěn na 4. straně obálky. Je možno použít i negativ filmu, získaný ofotografováním předlohy nakreslené ve větším měřítku. Můžete tak volit libovolný popis kláves a případně klávesy i barevně odlišit (acetonové modelářské barvy). Film je trvanlivý, ale pro první pokusy s mikropočítači vyhoví i papírový hmatník, ošetřený lakem. Díl *2* je oboustranně lepicí fólie. Díl *3* je fólie Alobal. Díl *4* je distanční vložka (obr. 3). Vložka je zhotovena také z oboustranně lepicí fólie. Díl *5* je deska s plošnými spoji podle obr. 4. Na rozdíl od běžné praxe nesmíme tu část desky, která nese kontaktní pole, natřít po zhotovení kalafunou. Na horním okraji desky je 13 vývodů klávesnice. Aby nebyla nutná oboustranná deska s plošnými spoji, musí se po dokončení klávesnice propojit holým pocínovaným drátem Cu, vedeným na spodní straně; 5 řad děr podle obr. 5.

Sestavení klávesnice vyžaduje tyto přípravné práce:

1. Musíme si zhotovit dvě oboustranně lepicí fólie pro díly *2* a *4* a to z jednostranně lepícího papíru „Pragofix". Vystřihneme z něj 4 kusy o rozměru 120 × 240 mm. Vždy dva kusy slepíme k sobě nelepicí stranou lepidlem Resolvan. Oproti návodu na obalu lepidla přiložíme k sobě strany, potřené lepidlem, ihned po jeho nanesení, aby bylo možno odstranit vzduchové bublinky mezi papíry. Po slepení archy zatížíme a necháme týden schnout. Papíry musí být perfektně slepeny po celé ploše!

2. Hmatník na obálce AR/B vystřihneme a přestříkáme bezbarvým lakem na nábytek (spray) v několika vrstvách. První vrstva musí být tenká, protože tiskárenská barva je rozpustná v ředidle laku. Další vrstvy je možno udělat tlustší. Každá vrstva by měla schnout tak dlouho, až není cítit ředidlem. Pak hmatník vystřihneme.

3. Jedna ze zhotovených oboustranně lepících fólií se upraví tak, aby vznikl díl *4* – distanční vložka. Vnější obrysy se zatím neupravují, pouze si je na fólii nakreslíme tužkou. Díry o Ø 12 mm se vyrazí výsečníkem.

4. Deska s plošnými spoji s kontakty klávesnice se ostřihne na rozměr 192,5 × 110 mm.

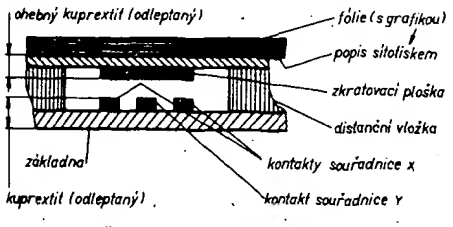
Jsou-li přípravné práce skončeny, můžeme začít se sestavou klávesnice:

1. Nejprve slepíme vzájemně hmatník (*1*), fólii (*2*) a Alobal (*3*). Ze zbylé oboustranně lepící fólie stáhneme krycí papír a nalepíme na ni hmatník. Lepit musíme postupně od jednoho kraje tak, aby nevznikly vzduchové bublinky. Lepené vrstvy už nelze od sebe oddělit a případné bublinky je možno odstranit jen propíchnutím jehlou! Podobným způsobem se na druhou stranu fólie nalepí vyhlazená fólie Alobal o rozměru asi 120 × 240 mm. Teprve po slepení celou podsestavu dílů *1, 2* a *3* ořízneme podle rohových značek na hmatníku (nejlépe nožem, podle ocelového pravítka).

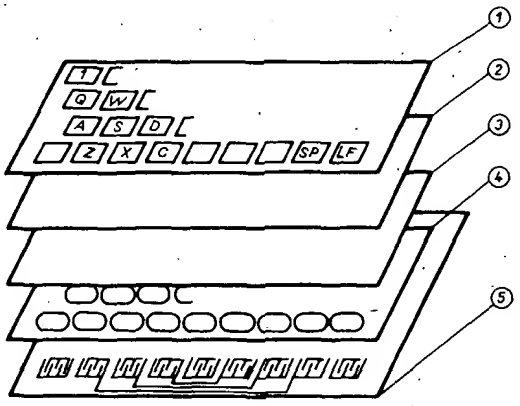
2. Dále slepíme podsestavu dílů *4* a *5*. Vystřihneme podle obrysu děrovanou distanční vložku *4*. Těsně před dalším postupem vyčistíme povrch měděné fólie kontaktů na desce s plošnými spoji jemným smirkovým plátnem (500, tzv. sépiový) a pak ošetříme Kontaktolem nebo podobným přípravkem. Pak stáhneme spodní krycí papír z distanční vložky *4* a nalepíme

Tab. 4. Typické parametry membránových klávesnic

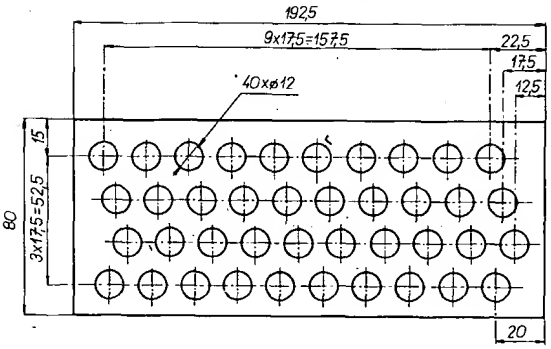
Okolní teplota	−20° až 65 °C.
Odolnost	proti prachu, vodě a oleji.
Zdvih	0,25 mm typ.
Kontaktní síla	1,5 N typ.
Doba života	5.10^6 sepnutí.
Napětí	0,5 až 30 V.
Proud	10 μA až 100 mA.
Přechodový odpor	1 Ω (kovové), 100 Ω (pasty).
Izolační odpor	10 MΩ.
Čas uklidnění kont.	10 ms typ.
Kapacita kontaktu	20 pF/kontakt.
Min. plocha pro hmatník	12 × 12 mm.
Zapojení spínačů	X–Y, nebo jeden společný vodič.



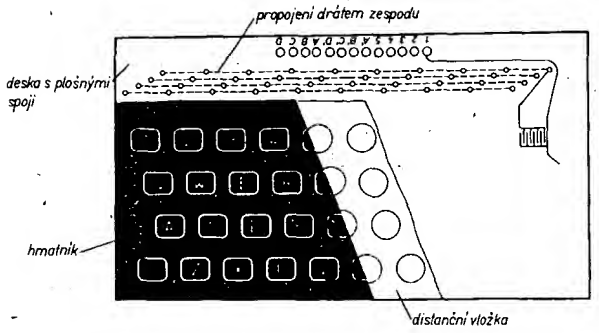
Obr. 1. Řez membránovou klávesnicí



Obr. 2. Díly membránové klávesnice; 1 – fólie s nápisy, 2 – oboustranně lepicí fólie, 3 – alobal, 4 – oboustranně lepicí fólie (obr. 3), 5 – deska s plošnými spoji (obr. 4)



Obr. 3. Distanční vložka z oboustranně lepící fólie – díl 4



Obr. 5. Sestava membránové klávesnice

ji na desku se spoji. Protože již máme desku se spoji *5* i vložku *4* upravené na konečný rozměr, je pro lepení referenční spodní a levý kraj vložky i desky se spoji. Při lepení desky se spoji pomůže přípravek, který si zhotovíme z rovné dřevěné desky a 4 hřebíčků. Hřebíčky zatlučeme těsně u krajů desky, dva dole a jeden při levé a pravé hraně, asi 10 mm od spodních rohů desky.

3. Poslední operací je slepení obou podsestav. Povrch Alobalu ošetříme opět Kontaktolem. Stáhneme krycí papír z horní strany distanční vložky *4* a slepíme v přípravku obě podsestavy dohromady. Celou sestavu můžeme pak přestříkat bezbarvým lakem, ale musíme zakrýt místa, na která budeme ještě pájet. Lakem nastříkáme i hrany klávesnice a tím budou papírové vložky chráněny proti vlhkosti. Klávesnici pak můžeme čistit vlhkým hadříkem a mýdlem.

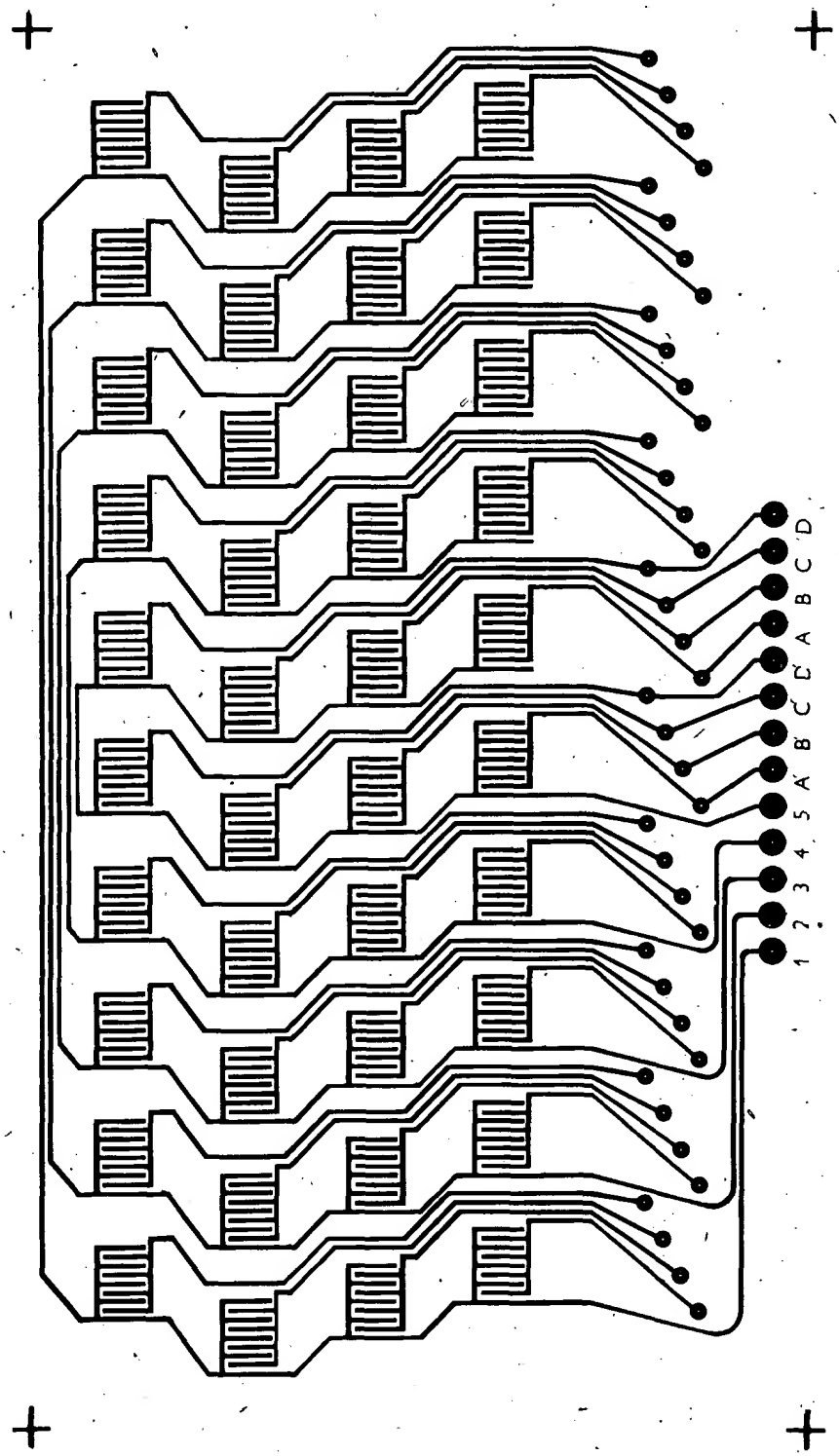
Nakonec propojíme řádky klávesnice podle sestavy na obr. 5 a ocínujeme pájecí plošky pro připojení přívodů. Klávesnici můžeme vyzkoušet ohmmetrem podle zapojení ve schématu ANK-1 (obr. 6).

ANK-1

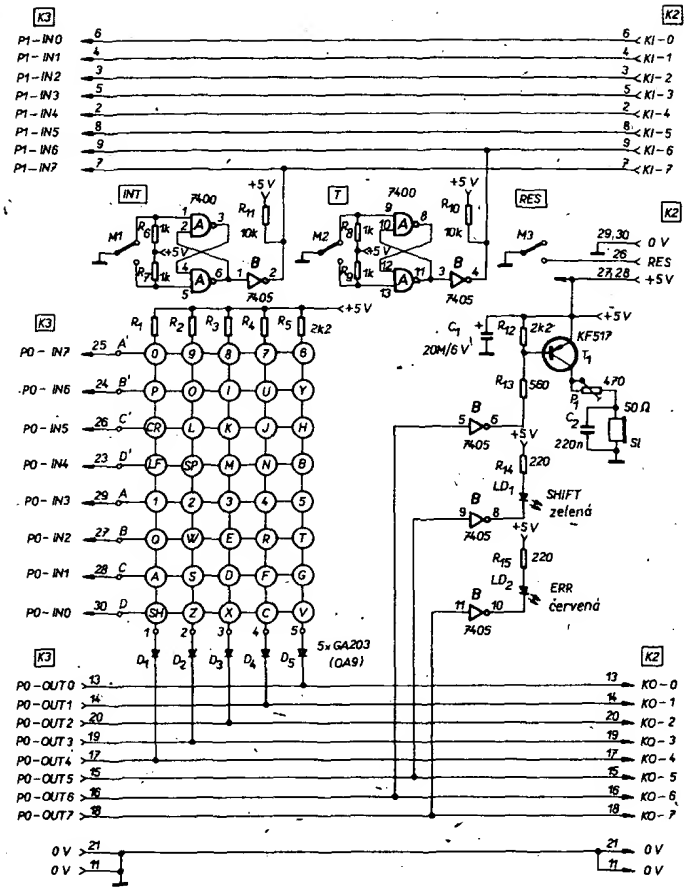
Klávesnice ANK-1 je připojena kabelem ke konektoru K2 desky JPR-1. Na konektor jsou vyvedeny vstupní porty P0 a P1 a výstupní port P0. Vyřešit všechny konektory a kabely mikropočítačového systému optimálně je problém, neboť dostupné konektory neumožňují „systémové" konstrukční řešení celého mikropočítače. Proto je klávesnice ANK-1 připojena kabelem prakticky ke třem portům. Vstupní port P0 je celý využit pro čtení logických úrovní z osmi výstupních vodičů membránové klávesnice. Výstupní port P0 má 5 bitů použitých pro „ohmatávání" sloupců klávesnice. Další 3 bity jsou použity pro ovládání dvou LED (ERR a SHIFT) a pro spínání telefonní vložky. Dioda SHIFT se v programech používá k indikaci stlačení tlačítka přeřazení na horní znaky klávesnice a dioda ERR slouží k libovolné indikaci (např. chyby obsluhy nebo chyby po testu pamětí atd.). Výstupní port P0 prochází klávesnicí ze vstupního konektoru K3 na výstupní konektor K2. Stejně tak prochází klávesnicí port P1, který je v klávesnici využit jen bity 7 a 6. Bit 7 přenáší informaci o stavu tlačítka INT a bit 6 o stavu tlačítka T.

Porty P0 – OUT a P1 – IN je možno použít pro různé pokusy s připojováním vstupů a výstupů k JPR-1. V systému JPR-1 počítáme s těmito porty pro připojení protahovacího snímače děrné pásky a pro připojení optického snímače proužkového kódu časopisu MC.

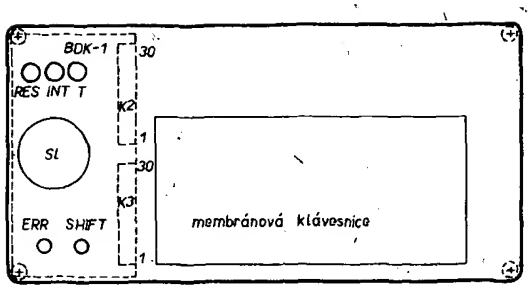
Celkové schéma zapojení klávesnice je na obr. 6. Výstupy portu P0 jsou připojeny k membránové klávesnici přes oddělovací germaniové diody D_1 až D_5. Tyto diody



Obr. 4. Deska s plošnými spoji membránové klávesnice



Obr. 6. Celkové schéma ANK-1

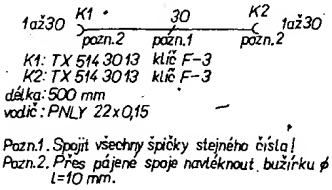


Obr. 7. Sestava ANK-1

JPR-1 a konektorem K3 ANK-1 je jednoduchý, protože je vlastně propojeno všech 30 špiček stejných čísel (obr. 10). I když nejsou všechny vodiče použity, vznikne kabel, který se může hodit pro testování nebo při vývoji dalších desek a zařízení. Kabel má označení KB-01.

Programování

Každé zařízení nebo deska, patřící do mikropočítačového systému, musí mít ve své dokumentaci základní údaje pro pro-



Obr. 8. Kabel KB-01 pro propojení JPR-1 a ANK-1

chrání výstupy obvodu 3212, který budí bity 0 až 4 portu P0. Stlačíme-li dvě tlačítka, a to je při přeřazení nutné, zkratujeme tím vlastně dva sloupce klávesnice. Odpory R_1 až R_5 zajišťují úroveň H na nevybraných sloupcích. Klopné obvody typu R-S odstraňují důsledky zákmitů kontaktů mikrospínačů M_1 a M_2. Invertory B/2 a B/4 budí svými výstupy s otevřeným kolektorem vstupy 7 a 6 portu P1. Otevřené kolektory umožňují využít portu P1, nejsou-li stisknuta tlačítka INT a T. Invertory B/8 a B/10 spínají LED a invertor B/6 budí tranzistorový spínač sluchátkové vložky. Potenciometrem P_1 můžeme nastavit hlasitost zvuku, který „vyrábí" počítač periodickým nastavováním bitu 6 výstupního portu P0. Změnou kapacity kondenzátoru C_2 lze měnit tón. Mikrospínač M_3 je vyveden společně s napájením na konektor K2 a propojuje se se systémovým konektorem na jednotce sběrnice a zdroje. Tlačítkem RES pak můžeme generovat signál RTL, který je na špičce 1 konektorů sběrnice ARB-1.

Klávesnice ANK-1 je vestavěna do krabičky od hry LOGIK, kterou vyrábí družstvo Maják. V rozích dna krabičky jsou přišroubovány 4 pryžové nožky, aby bylo možno vést kabel pod klávesnicí. Logika klávesnice je postavena na části univerzální desky BDK-1. V přepážce krabičky jsou vyříznuty dva otvory pro konektory K2 a K3. Číslování těchto konektorů je zachováno z desky BDK-1. Mikrospínače jsou ovládány tlačítky přes fosforbronzové planžety, připevněné nad mikrospínači. Sestava ANK-1 je na obr. 7. V tab. 5 je zapojení konektoru K2 a v tab. 6 konektoru K3. Kabel mezi konektorem K2 na

Tab. 5. Zapojení konektoru K2 klávesnice ANK-1

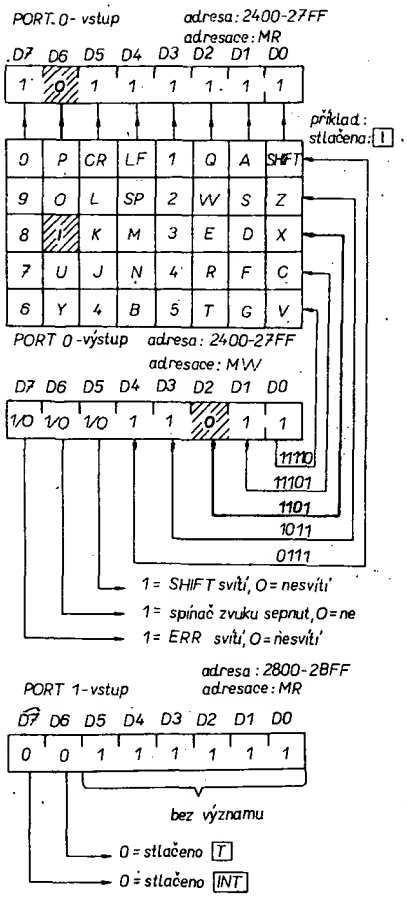
Č.	Signál	Název	Typ	Č.	Signál	Název	Typ
1				2	KI-4		
3	KI-2			4	KI-1	vstupy kláves-	
5	KI-3	vstupy kláves-		6	KI-0	nice do portu P1	
7	KI-7	nice do portu		8	KI-5		
9	KI-6			10			
11	0 V	zem		12			
13	K0-0			14	K0-1		
15	K0-5	výstupy kláves.		16	K0-6	výstupy kláves	
17	K0-4	z portu P0		18	K0-7	z portu P0	
19	K0-3			20	K0-2		
21	0 V	zem		22			
23				24			
25				26	RES	tlačítko RESET	
27	+5 V	napájení		28	+5 V	napájení	
29	0 V	zem		30	0 V	zem	

Číslo konektoru:	K2	Konektor:	TY 513 30 11
Deska zařízení:	ANK-1	Protikus:	TX 514 30 13
Klíčování:	C-6		

Tab. 6. Zapojení konektoru K3 klávesnice ANK-1

Č.	Signál	Název	Typ	Č.	Signál	Název	Typ
1				2	P1-IN4		
3	P1-IN2			4	P1-IN1	vstupy port 1	
5	P1-IN3	vstupy port 1		6	P1-IN0		
7	P1-IN7			8	P1-IN5		
9	P1-IN6			10			
11	0 V	zem		12			
13	P0-OUT0			14	P0-OUT1		
15	P0-OUT5	výstupy port 0		16	P0-OUT6	výstupy port 0	
17	P0-OUT4			18	P0-OUT7		
19	P0-OUT3			20	P0-OUT2		
21	0 V	zem		22			
23	P0-IN4			24	P0-IN6		
25	P0-IN7	vstupy port 0		26	P0-IN5	vstupy port 0	
27	P0-IN2			28	P0-IN1		
29	P0-IN3			30	P0-IN0		

Číslo konektoru:	K3	Konektor:	TY 513 30 11
Deska zařízení:	ANK-1	Protikus:	TX 514 30 13
Klíčování:	F-3		



Obr. 9. Programování ANK-1 s JPR-1 (1 = log. 1,0 = log. 0)

gramování. Podle nich pak může programátor psát aplikační programy. Na obr. 9 jsou významy jednotlivých bitů při spolupráci JPR-1 a ANK-1.

Seznam součástek desky ANK-1

Odpory (TR 191 ±10 %)

R_1 až R_5, R_{12}	2,2 kΩ
R_6 až R_9	1 kΩ
R_{10}, R_{11}	10 kΩ
R_{13}	560 Ω
R_{14}, R_{15}	220 Ω

Kondenzátory

C_1	20 µF/6 V, TE 981
C_2	0,1 až 1 µF, TC 215

Polovodičové součástky

A	MH7400
B	MH7405
T_1	KF517
D_1 až D_5	Ge diody (GA203, OA9)
LD_1	zelená dioda LED, VQA23 (NDR)
LD_2	červená dioda LED, LQ114

Ostatní součástky

mikrospínače	M_1 až M_3 B593
P_1	470 Ω, TP 095
SI	sluchátková vložka 50 Ω
K2, K3	konektory FRB TY 513 30 11
deska s plošnými spoji BDK-1	
membránová klávesnice podle návodu, krabička od hry LOGIK, 4 pryžové nožičky	
kabel	KB-01, 2 ks, konektor FRB TX 514 30 13, 1 m vodiče PNLY 22 × 0,15 mm; 0,3 m bužírky o Ø 2 mm

DESKA PAMĚTÍ REM-1

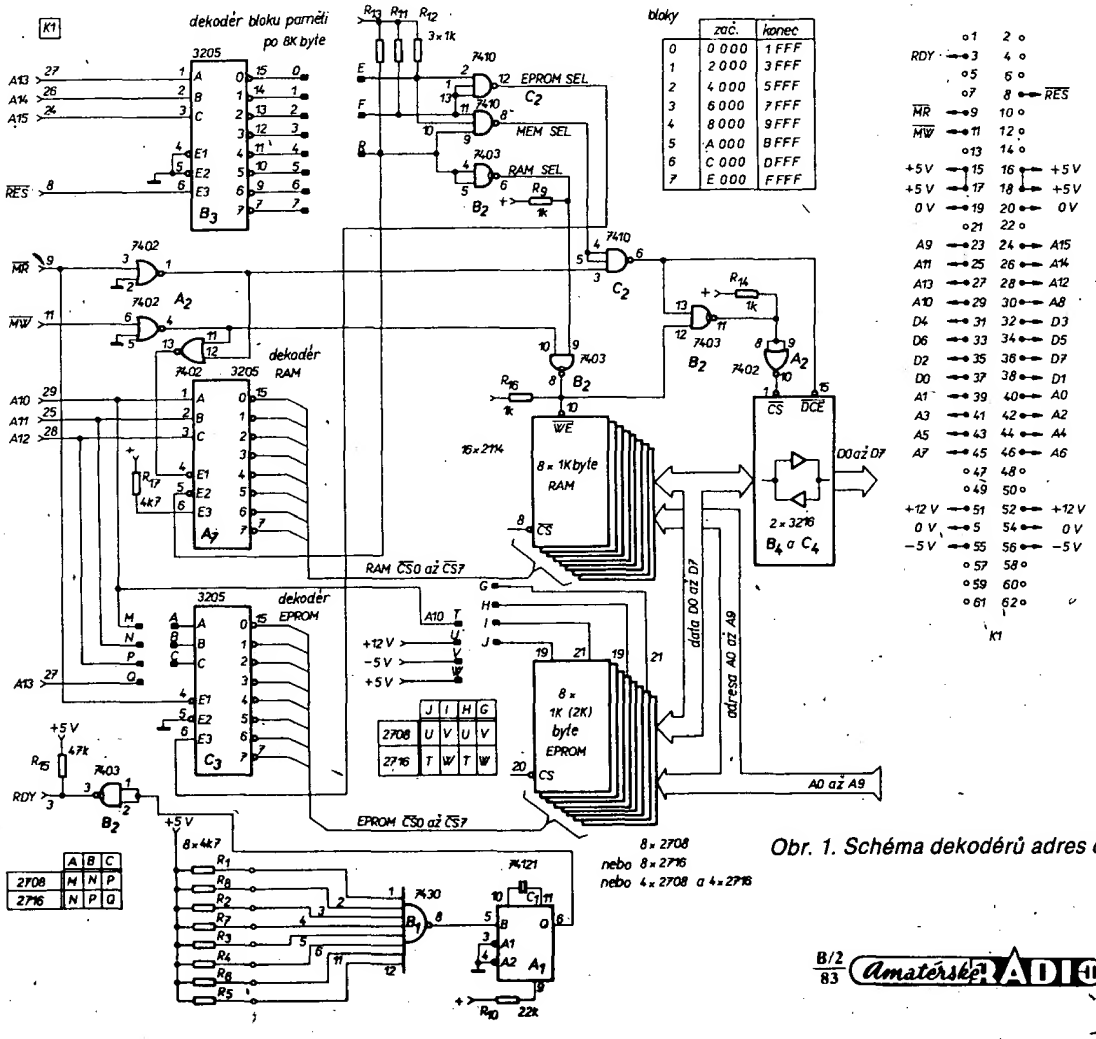
Zkratka REM znamená *R*AM a *E*PROM *M*emory, neboli část této desky obsahuje paměť RAM z obvodů 2114 a druhá část paměť EPROM z obvodů 2708 nebo 2716.

Jak již bylo řečeno při popisu desky procesoru, není možné, aby paměťové obvody na desce JPR-1 pracovaly s velkou zátěží sběrnice ARB-1. Chceme-li mít větší systém s více deskami, je nutné obvody RAM a EPROM vyjmout z desky procesoru JPR-1 a paměť systému realizovat deskou REM-1. Deska REM-1 má zesilovače datových signálů, které oddělují výstupy paměťových obvodů od sběrnice. Deska REM-1 má při plném osazení obvody kapacitu 8K RAM a 16K EPROM.

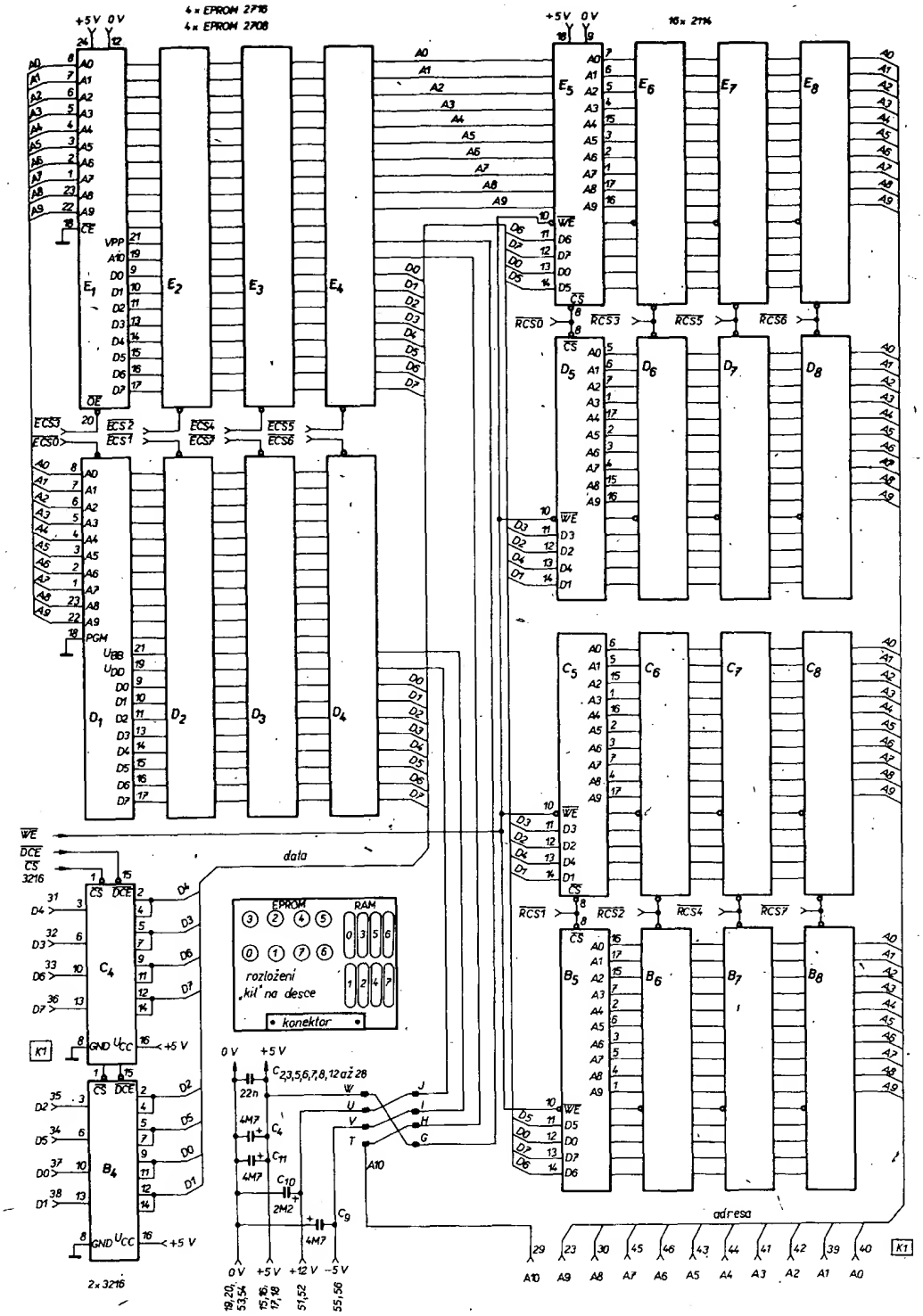
Popis činnosti

Schéma desky REM-1 je na obr. 1 a obr. 2. Na obr. 1 jsou detailně rozkresleny obvody adresace, které jsou u desek pamětí často složitější než vlastní „pole" paměťových obvodů. Na obr. 2 je propojení pamětí, které je však zajímavé pouze při oživování nebo hledání závad. Nebudu se zde zabývat principem činnosti pamětí RAM a EPROM, protože tato problematika již byla na stránkách AR vysvětlena.

Navrhujeme-li desku pamětí, jsme postaveni před otázku, jaké možnosti dát uživateli pro „posazení" pamětí do celkového adresového prostoru mikropočítačového systému. Jedna krajní možnost je zvolit pevnou adresu. Toto řešení je možné pouze u jednoúčelových systémů. Druhá krajní možnost je umožnit co nejširší volbu adresy. Vzhledem k malé desce systému JPR-1 jsem musel zvolit určitý kompromis. Velké množství propojek pro volbu adres zabírá na desce hodně místa a také počet obvodů se rychle zvětšuje s univerzálností volby adresového prostoru pamětí RAM a EPROM. Proto jsem zvolil jako základní dělení bloky paměti s kapacitou 8K.



Obr. 1. Schéma dekodérů adres desky REM-1



Obr. 2. Schéma propojení pamětí (kresleno pro E_1 až E_4 – 2716, D_1 až D_4 – 2708) desky REM-1

První stupeň dekodéru adresy tvoří obvod B_3 na obr. 1. Vstupy A, B, C dekodéru 3205 jsou připojeny na tři nejvyšší adresové bity sběrnice. Kladný povolovací vstup E3 je připojen na signál $\overline{RES}$, který blokuje výběr jakékoli paměti při zapnutí systému. Výstupy dekodéru B_3 jsou vedeny na propojky 0 až 7. Tyto propojky je možno spojit se špičkami označenými E, F a R. Hradlo $C_2/12$ (obvod C_2 – výstup hradla – špička 12) pracuje ve funkci NEBO. Špičku E nebo F je možno spojit s body 0 až 7, a protože má dekodér B3 aktivní výstupní úrovně L, způsobí jakákoli úroveň L na vstupu hradla $C_2/12$ na výstupu úroveň H a tím vznik signálu EPROM SEL. Při použití pamětí 1K (2708) je samozřejmě možné použít pouze jednu propojku, neboť jeden výstup dekodéru „hlídá" adresový prostor o kapacitě 8K a víc nejde s obvody 2708 realizovat! Při použití obvodů 2716, které mají dvojnásobnou kapacitu, je nutno použít obě špičky, E i F, a zvolit tak adresový prostor o kapacitě 16K. Pak není nutné, aby bloky následovaly za sebou a je možno zvolit první blok v rozsahu 0000 až 1FFF a druhý třeba v rozsahu 6000 až 7FFF. Je-li aktivní signál EPROM SEL, je aktivován i dekodér EPROM, tvořený obvodem C_3. Opět je použit obvod 3205, který je pro dekódování adresy velice výhodný, ať již tím, že má malý vstupní proud, nebo tím, že má více povolovacích vstupů než obvod 7442. Dekodér C_3 má na vstupy A, B a C přivedeny adresy A10, A11 a A12, nebo adresy A11, A12 a A13. Výstupy tohoto dekodéru pak vybírají ze zvoleného bloku o kapacitě 8K (16K) oblasti buď 1K nebo 2K (nebo stránky) adres. Výstupy dekodéru C_3 pak přímo aktivují jednotlivé čipy pamětí EPROM. Jelikož se jedná o paměti, z nichž procesor pouze čte, je dekodér C_3 aktivní pouze při signálu $\overline{MR}$ z procesoru. Pro aktivaci je využit vstup E1 obvodu C_3. Budete-li mít na desce „směs" obvodů 2708 a 2716, nezapomeňte na to, že u pamětí 1K se objeví stejný obsah ve 2K po sobě, podobně jako u JPR-1.

Adresový prostor pro paměť RAM je volen spojením špičky R s výstupem dekodéru B_3. Hradlo $B_2/6$ pracuje pouze jako invertor. Signál REM SEL povoluje zápis do pamětí RAM (hradlo $B_2/8$). Objeví-li se úroveň L na špičce R, je aktivován navíc dekodér jednotlivých „kil" paměti RAM (obvod A_7) a je generován signál MEM SEL.

Dekodér A_7 je aktivní při signálech $\overline{MR}$ a $\overline{MW}$ a jeho výstupy ovládají výběr jednotlivých „kil" pamětí RAM. Hradla $A_2/1$ a $A_2/4$ pracují pouze jako oddělovače, aby deska minimálně zatěžovala řídicí signály sběrnice. Hradlo $A_2/13$ pracuje jako obvod NEBO a jeho výstup aktivuje dekodér A_7.

Signál MEM SEL je aktivní při jakékoli úrovni L na špičkách E, F nebo R a je vlastně nejdůležitějším signálem desky. Je-li tento signál aktivní, pak při čtení z paměti (aktivní i MR) je zvolen směr budičů B_4 a C_4 sběrnice ven, k procesoru. Je třeba si uvědomit, že jednoduchost dekodérů adres způsobí, že paměť se na sběrnici hlásí i tehdy, nemáme-li všechny objímky osazené obvody. Není tedy možné nechat třeba na desce procesoru jednu paměť a na desce REM-1 druhou ve stejném bloku 8K! Kdyby někdo potřeboval dělat podobné pokusy, musel by využít volných vstupů dekodéru B_3 (E_1 a E_2).

Hradlo $B_2/11$ aktivuje obvody B_4 a C_4 buď při zápisu do paměti RAM, nebo při čtení z jakékoli paměti.

Hradlo $B_1/8$ tvoří obvod NEBO pro signály CS jednotlivých pamětí EPROM. Vstupy tohoto hradla je nutno propojit drátky. S využitím tohoto hradla je totiž počítáno jen v nejkrajnějším případě: společně s monostabilním obvodem A_1 a hradlem $B_2/3$ vytváří obvod pro vznik žádosti o „počkání" tehdy, bude-li na desce pomalejší paměť EPROM. Ve většině případů není třeba obvody A_1 a B_1 vůbec osazovat. Nebude-li na desce obvod A_1, je ovšem nutné uzemnit vývod 6, aby byl výstup hradla $B_2/3$ na úrovni H.

Druhá část schématu (obr. 2) znázorňuje vlastně provedení desky s plošnými spoji. U pamětí typu RAM není nutné dodržet správné číslování adresových ani datových vývodů. Propojení těchto pamětí je proto podřízeno jednoduchosti plošných spojů. U pamětí EPROM je nutné zapojení vývodů dodržet!

Aby bylo možno použít paměti EPROM 1K i 2K, je deska REM-1 (obr. 3 až 5) navržena pro oba typy. Typ paměti se volí propojkami. Paměti EPROM jsou na desce ve dvou řadách a v jedné řadě mohou být paměti pouze jednoho typu. Na propojky je nutné dát velký pozor a je vůbec nejlepší rozhodnout se pouze pro jeden typ a desku označit upozorněním, o jaký typ se jedná. Pozor také na to, že firma TI dělala jeden čas paměti 2K s napájením jako 2708! Uprostřed obr. 2 je schematicky znázorněno rozložení jednotlivých obvodů podle adresace. Toto rozložení vyšlo z návrhu plošných spojů.

Tab. 1. Mapa paměti po blocích 1K

Adresové bity A15, A14 a A13	Adresové bity A12, A11 a A10							
	000	001	010	011	100	101	110	111
000	BLOK 0 0000 –03FF	BLOK 1 0400 –07FF	BLOK 2 0800 –0BFF	BLOK 3 0C00 –0FFF	BLOK 4 1000 –13FF	BLOK 5 1400 –17FF	BLOK 6 1800 –1BFF	BLOK 7 1C00 –1FFF
001	BLOK 8 2000 –23FF	BLOK 9 2400 –27FF	BLOK 10 2800 –2BFF	BLOK 11 2C00 –2FFF	BLOK 12 3000 –33FF	BLOK 13 3400 –37FF	BLOK 14 3800 –3BFF	BLOK 15 3C00 –3FFF
010	BLOK 16 4000 –43FF	BLOK 17 4400 –47FF	BLOK 18 4800 –4BFF	BLOK 19 4C00 –4FFF	BLOK 20 5000 –53FF	BLOK 21 5400 –57FF	BLOK 22 5800 –5BFF	BLOK 23 5C00 –5FFF
011	BLOK 24 6000 –63FF	BLOK 25 6400 –67FF	BLOK 26 6800 –6BFF	BLOK 27 6C00 –6FFF	BLOK 28 7000 –73FF	BLOK 29 7400 –77FF	BLOK 30 7800 –7BFF	BLOK 31 7C00 –7FFF
100	BLOK 32 8000 –83FF	BLOK 33 8400 –87FF	BLOK 34 8800 –8BFF	BLOK 35 8C00 –8FFF	BLOK 36 9000 –93FF	BLOK 37 9400 –97FF	BLOK 38 9800 –98FF	BLOK 39 9C00 –9FFF
101	BLOK 40 A000 –A3FF	BLOK 41 A400 –A7FF	BLOK 42 A800 –ABFF	BLOK 43 AC00 –AFFF	BLOK 44 B000 –B3FF	BLOK 45 B400 –B7FF	BLOK 46 B800 –BBFF	BLOK 47 BC00 –BFFF
110	BLOK 48 C000 –C3FF	BLOK 49 C400 –C7FF	BLOK 50 C800 –CBFF	BLOK 51 CC00 –CFFF	BLOK 52 D000 –D3FF	BLOK 53 D400 –D7FF	BLOK 54 D800 –DBFF	BLOK 55 DC00 –DFFF
111	BLOK 56 E000 –E3FF	BLOK 57 E400 –E7FF	BLOK 58 E800 –EBFF	BLOK 59 EC00 –EFFF	BLOK 60 F000 –F3FF	BLOK 61 F300 –F7FF	BLOK 62 F800 –FBFF	BLOK 63 FC00 –FFFF

Tab. 2. Zapojení konektoru K1 desky REM-1

Č.	Signál	Název	Typ	Č.	Signál	Název	Typ
1				2			
3	RDY	READY	OUT	4			
5				6			
7				8	$\overline{RES}$		INP
9	$\overline{MR}$	čtení z paměti	INP	10			
11	$\overline{MW}$	zápis do paměti	INP	12			
13				14			
15	+5 V	napájení	NAP	16	+5 V	napájení	NAP
17	+5 V		NAP	18	+5 V		NAP
19	0 V	zem	NAP	20	0 V	zem	NAP
21				22			
23	A9	adresa	INP	24	A15	adresa	INP
25	A11		INP	26	A14		INP
27	A13		INP	28	A12		INP
29	A10		INP	30	A8		INP
31	D4	data	BD	32	D3	data	BD
33	D6		BD	34	D5		BD
35	D2		BD	36	D7		BD
37	D0		BD	38	D1		BD
39	A1	adresa	INP	40	A0	adresa	INP
41	A3		INP	42	A2		INP
43	A5		INP	44	A4		INP
45	A7		INP	46	A6		INP
51	+12 V	napájení	NAP	52	+12 V	napájení	NAP
53	0 V	zem	NAP	54	0 V	zem	NAP
55	–5 V	napájení	NAP	56	–5 V	napájení	NAP

Číslo konektoru: K1 | Konektor: TY 517 62 11
Deska/zařízení: REM-1 | Protikus: TX 518 62 12
Klíčování: C-6

INP – vstup
BD – obousměrný
OUT – výstup
NAP – napájení

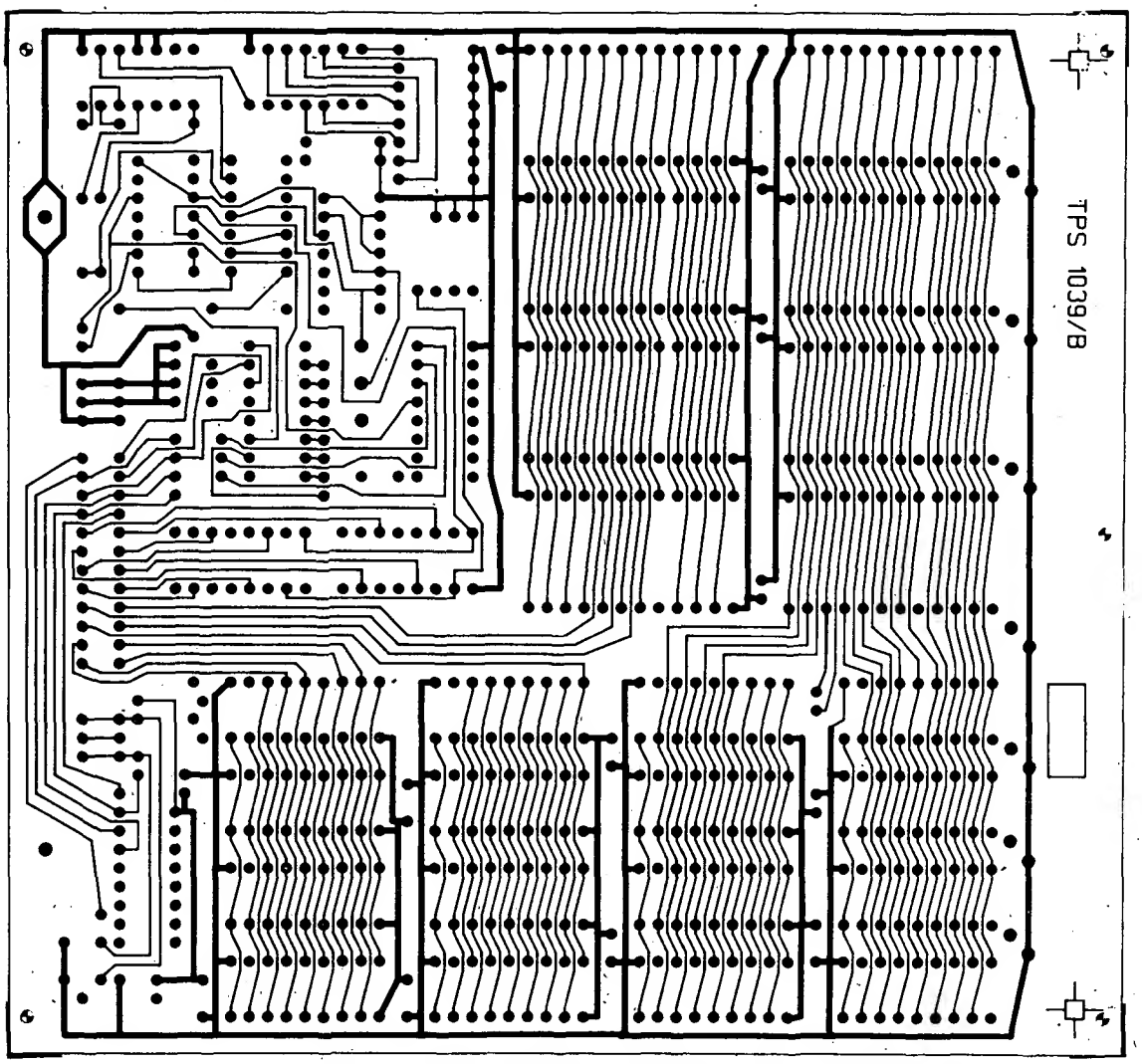
Oživení desky REM-1

Desku oživíme jednoduše přípravkem TST-03 a logickou sondou. Nejjistější, i když nejpracnější, je změřit všechny adresové vstupy všech obvodů přímo na jejich objímkách. Přepínáme jeden adresový bit na přípravku a měříme, zda se mění všude, kde má být. Tímto postupem však nezjistíme případné zkraty mezi spoji na desce. Nejlepší je zhotovit si vícenásobnou sondu se šestnácti, osmnácti a dvaceti čtyřmi LED a zakončit ji kabelem, který zasuneme do objímek místo obvodu. Diody mohou indikovat i přítomnost napájecích napětí. Pomocí těchto sond proměříme desku velice rychle a najdeme i zkraty.

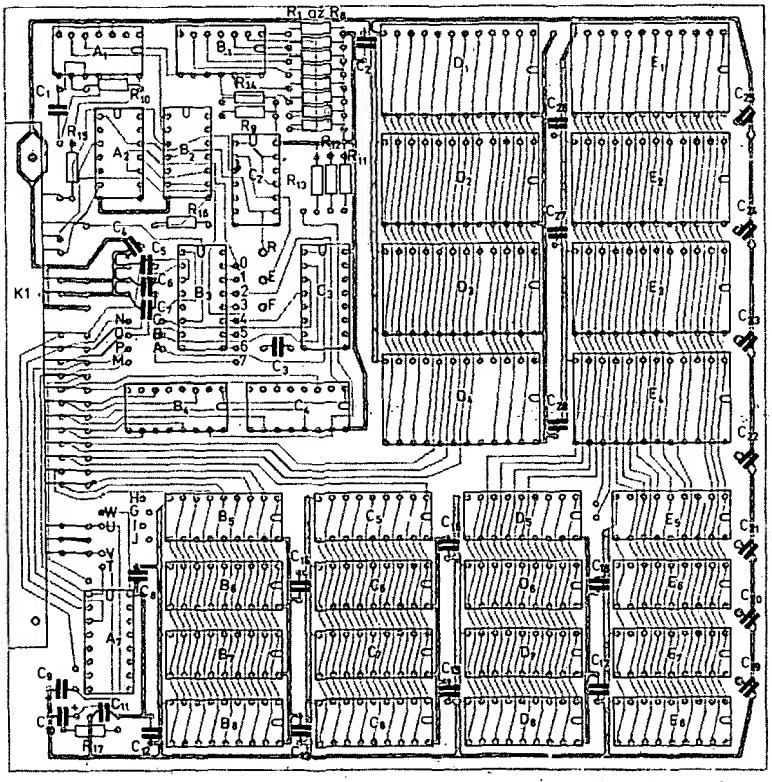
Dále změříme funkce jednotlivých dekodérů adres A_7, B_3 a C_3. Potom zhotovíme propojky tak, jak je budeme potřebovat a změříme signál $\overline{WE}$ ($8/B_2$), $\overline{CS}$ obvodů 3216 ($1/B_4$ a $1/C_4$) a $\overline{DCE}$ těchto obvodů ($15/B_4$ a $15/C_4$).

Pro ověření datových signálů pamětí EPROM použijeme přípravek TST-02. Pomocí přípravku, jehož kabel nasuneme místo jedné paměti EPROM, změříme, zda se správně čtou různé kombinace dat (FF, AA, 00 atd.). Čtení se na přípravku TST-03 realizuje tlačítkem MR.

Potom změříme napájecí napětí na objímkách pamětí RAM a po zasunutí obvodů 2114 vyzkoušíme zapsat a číst z předvolených adres pomocí přípravku TST-03. Navolíme pomocí adresových přepínačů správnou adresu příslušného „kilového" bloku, nastavíme datovými přepínači data a stlačíme tlačítko MW. Potom můžeme ještě stejným postupem psát na další adresy. Pak se vrátíme na počáteční adresu a obsahy zkusíme pomocí tlačítka MR přečíst. Nefunguje-li nějaký bit, využijeme toho, že po dobu stlačení tlačítka MW jsou na datových vývodech pamětí 2114 data a přeměříme je sondou. Při pečlivém



Obr. 4. Deska s plošnými spoji REM-1, strana součástek



Obr. 3. Rozložení součástek na desce REM-1

proměření desky ještě před zasunutím obvodů by však neměly nastat podobné potíže.

Před zasunutím pamětí EPROM přeměříme znovu napájecí napětí a přesvědčíme se, zda je deska předvolena propojkami právě pro ten typ, který chceme použít.

Potom již můžeme otestovat desku pomocí programu v systému JPR-1. Před zasunutím desky REM-1 do sběrnice ARB-1 prověříme, zda adresy, které jsou na ní navoleny, nejsou již v systému použity, třeba na desce JPR-1.

Programování desky REM-1

Pro programátora je důležité pouze to, jaké adresy mají paměti a jaké pozice jsou na desce skutečně osazeny obvody. Pro lepší orientaci v adresovém prostoru systému JPR-1 je v tab. 1 mapa paměti po blocích 1K. Tabulka je dobrým pomocníkem při zapojování propojek na desce REM-1. Zapojení konektoru desky REM-1 je v tab. 2.

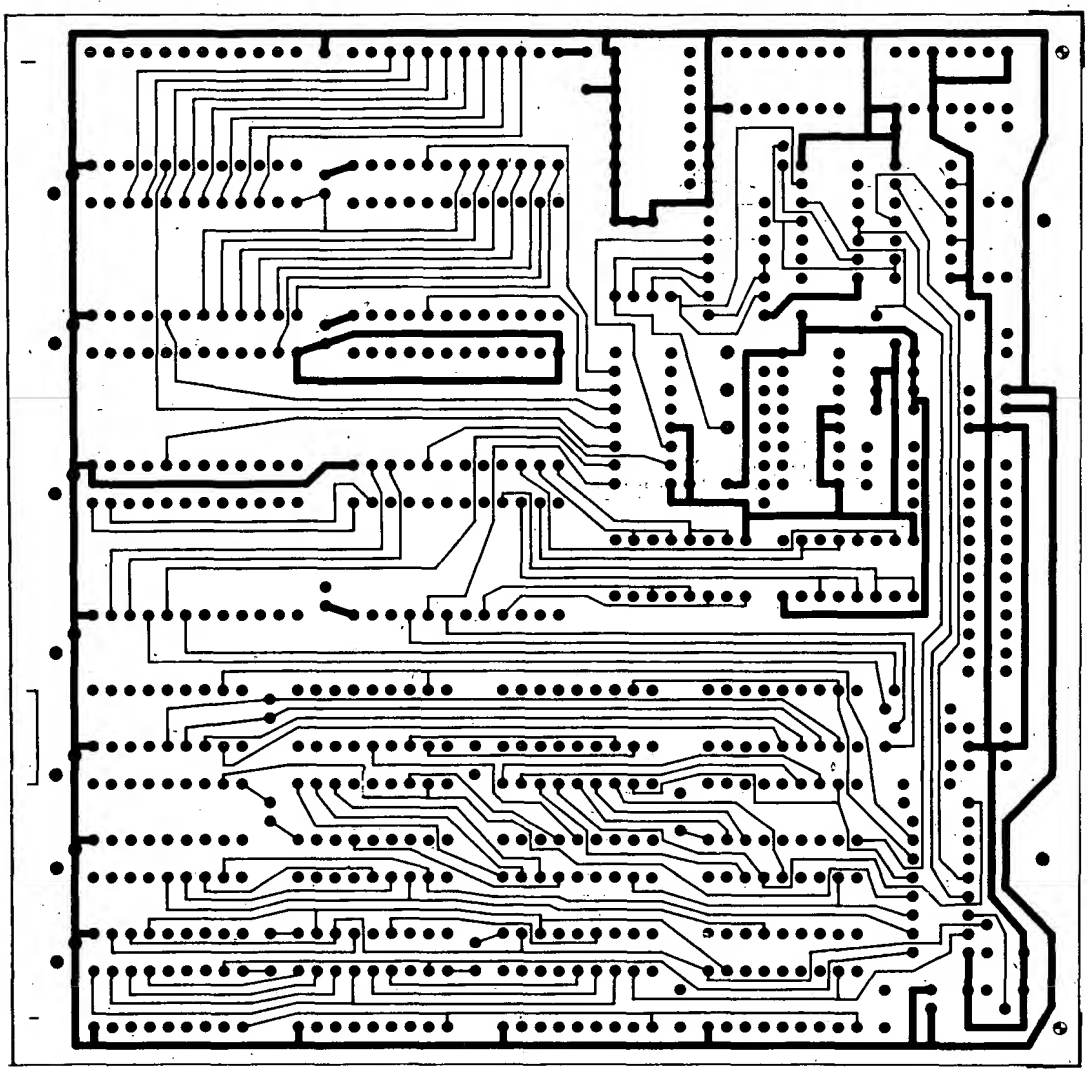
Seznam součástek

Odpory (TR 191)

R_1 až R_8, R_{17}	4,7 kΩ
R_9, R_{11} až R_{14}, R_{16}	1 kΩ
R_{10}	22 kΩ
R_{15}, R_{17}	47 kΩ

Kondenzátory

C_1	50 až 150 pF (viz text)
C_2, C_3, C_5 až C_8, C_{12} až C_{28}	22 nF, TK 783



Obr. 5. Deska s plošnými spoji REM-1, spodní strana

C_4, C_9, C_{11}	4,7 µF/6,3 V
C_{10}	2,2 µF/16 V
	–
Integrované obvody	
A_1	UCY74121
A_2	UCY7402
B_1	MH7430
B_2	MH7403
C_1	–
C_2	MH7410
A_7, B_3, C_3	MH3205
B_4, C_4	MH3216
E_5 až E_8, D_5 až D_8, C_5 až C_8, B_5 až B_8	RAM 2114
E_1 až E_4, D_1 až D_4	EPROM 2708 (2716)
K1	konektor FRB TY 517 62 11
objímka pro IO 24 vývodů, 8 ks	
objímka pro IO 18 vývodů, 16 ks	
deska s plošnými spoji REM-1	

ALFANUMERICKÝ DISPLEJ AND-1

Nejrozšířenějším komunikačním prostředkem mezi počítačem a člověkem je alfanumerický displej, využívající k zobrazení znaků televizní obrazovku. Tato zařízení, známá pod názvem „CRT display", umožňují poměrně jednoduše a levně zobrazit velké množství znaků nebo dokonce i grafických symbolů či obrázků. Doplní-li se displej o klávesnici a logiku, která je schopna přenášet oboustranně data, vznikne koncové zařízení počítače, tzv. *terminál*. Velká potřeba terminálů v bankách, obchodě a dopravě ovlivnila i vývoj mikroprocesorů. Obvod 8008, první osmibitový mikroprocesor firmy INTEL, byl prý vyvinut na základě objednávky firmy Datapoint, která jej potřebovala do svých terminálů.

Díky tomu, že je televizní přijímač nejdostupnějším zařízením pro každého zájemce o mikropočítače, používá se dnes u malých systémů jako základní vybavení. Velké systémy používají buď speciální obrazovkové terminály nebo barevné monitory. Chceme-li používat mikropočítač pro vývoj programů nebo pro aplikace, v nichž je komunikace s obsluhou řízeného stroje složitější, musíme vlastně vývojem alfanumerického displeje začít. Teprve podle možností displeje můžeme psát programové vybavení mikropočítačového systému. Každý program je závislý na počtu znaků, které displej zobrazí na jednom řádku i na počtu řádků a výběru znaků. Proto i systém JPR-1 potřeboval displej, který by nebyl ani špatný, ani složitý.

Začínáme-li vývoj nového zařízení, je třeba určit základní parametry, vlastně soubor cílů, kterých chceme dosáhnout. U displeje, který dostal pracovní název AND-1, jsem chtěl dosáhnout těchto cílů:

1. Použít paměť o kapacitě 1K byte, kterou lze sestavit ze dvou obvodů 2114.
2. Použít pro zobrazení TV přijímač nebo pro aplikace v průmyslu monitor AZJ462, který vyrábí TESLA Orava.
3. Vyřešit problém vystředění textové stránky na stínítku, neboť se velmi často stává, že stejný počítač „kreslí" na různých přijímačích s přesností polohy ±1 cm.
4. Využít generátoru znaků MHB2500, který již několik let je ve výrobě a není tak využíván, jak by si dobrá moderní součástka zasluhovala.

První úvahy každého, kdo navrhuje displej, začínají u formátu zobrazení dat na stínítku obrazovky a potom už lze přemýšlet o časové základně a dalších obvodech displeje.

Formát dat na stínítku obrazovky

Jedním ze základních parametrů displejů a terminálů je formát dat na stínítku obrazovky. Pokud je displej pouze alfanumerický, udává se formát dat počtem znaků, které se vejdou na jeden řádek a počtem řádků. Světovým standardem je 80 znaků na řádek a 24 řádků, tj. 1920 znaků na stínítku. Je to dáno tím, že 80 znaků je kapacita jednoho děrného štítku a proto jsou na 80 znaků uživatelé výpočetní techniky zvyklí. Pro kapacitu 80 znaků na jeden řádek jsou vyráběny i tiskárny a velmi často to bývala délka jedné „věty" při zpracování informací nebo dél-

ka bloku při záznamu na magnetickou pásku.

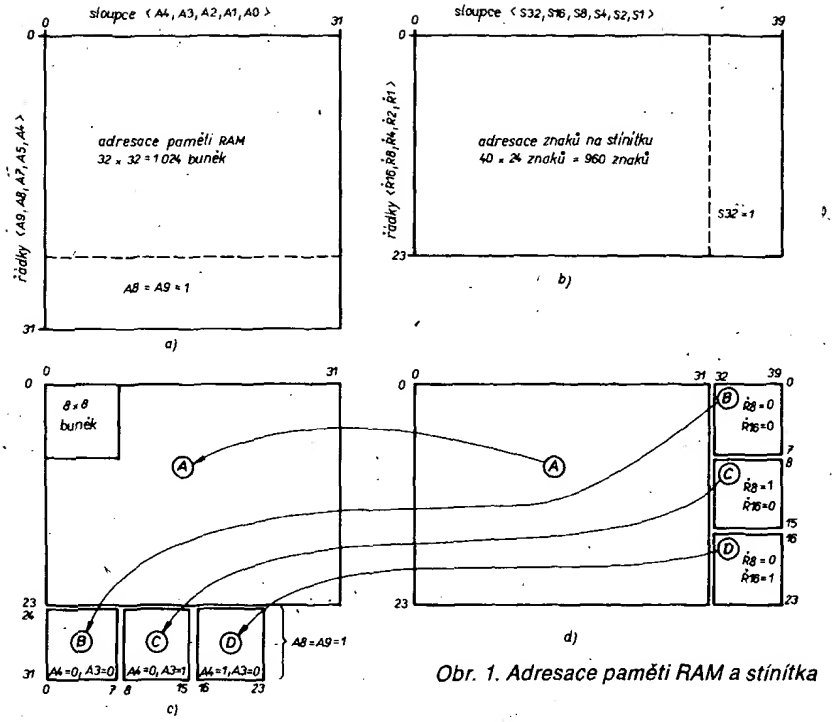
Počet znaků na řádek je však u displeje omezen kvalitou videomonitoru, zejména šířkou pásma a ztrátovými časy (návrat zatemněného paprsku zpět na začátek nového řádku). Chceme-li použít jako monitor běžný TV přijímač, musíme počítat s tím, že má šířku pásma asi 6 MHz, a že použitelná část periody řádků je pouze asi 40 μs, protože obrazovky v našich TV přijímačích nekreslí ostře v rozích stínítka. Tato omezení vedou často k tomu, že se používá buď 32 nebo 64 znaků na řádek. Při počtu 64 znaků na jednom řádku a šesti bodech na jeden znak potřebujeme však příliš vysoký kmitočet, abychom „stihli" řádek za 40 μs. Také nároky na přístupové časy paměti RAM a generátoru znaků ROM se neúměrně zvětší, protože je nutné vždy po 6 bodech znovu přečíst obsah obou pamětí. Tyto problémy lze řešit zařazením vyrovnávacích registrů mezi paměti, zvětší se však nároky na počet součástek a jejich rychlost.

U systému JPR-1 jsem byl omezen velikostí desky – proto bylo nutné s počtem součástek šetřit. Protože je u velké řady mikropočítačů použit formát poloviční, vzhledem ke standardu, zvolil jsem jako cíl vývoje 40 znaků na řádek. Počet řádků pak vyšel z toho, že jsem chtěl použít kapacitu paměti RAM 1K byte. Nakreslit 25 řádků znaků na stínítko obrazovky TV přijímače je však také problém. Záleží jednak na tom, kolika prázdnými TV řádky chceme řádky znaků od sebe oddělit, a jednak opět na rychlosti návratu paprsku a ostrosti kresby v rozích obrazovky.

Základním parametrem tedy bylo 40 znaků na řádek. Dále bylo nutné přemýšlet o tom, jak adresovat paměť RAM v závislosti na adrese znaku na stínítku obrazovky. Adresa paměti 1K byte je dána binárním číslem o délce 10 bitů. Adresa polohy znaku v řádku je dána číslem 0 až 39 a na vyjádření tohoto čísla potřebujeme 6 bitů. Adresa znakového řádku je dána číslem 0 až 24 a na vyjádření tohoto čísla potřebujeme 5 bitů. Celkem je tedy tvořena adresa polohy znaku na stínítku číslem o délce 11 bitů. Nechceme-li, aby programátor musel polohu znaku kódovat složitě a chceme-li pro čítače polohy použít binární čítače, je nutné překódovat adresu polohy na stínítku (11 bitů) na adresu paměti 1K byte (10 bitů).

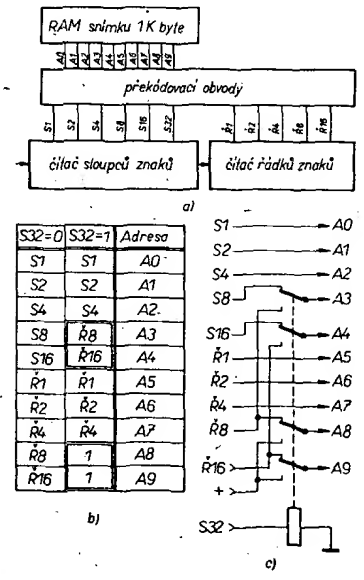
Podívejme se nyní na obr. 1a, kde je adresová mapa paměti 1K nakreslena jako čtverec 32×32 adres. Sloupce paměti určují adresové bity A0 až A4 (5 bitů) a řádky bity A5 až A9 – také 5 bitů. Na obr. 1b je adresová mapa znaků, zobrazovaných na stínítku, nakreslená jako obdélník 40×24. Sloupce určují adresové bity S, v nichž je číslo, vyjadřující jejich binární váhu, S1, 2, 4 až 32. Znakové řádky jsou určeny adresovými bity Ř1 až Ř16. Z následujících úvah nyní vyplyne postup řešení daného problému a také důvod, proč není řádků 25, ale jen 24.

Podíváme-li se na obě mapy, vidíme, že na jedné něco přebývá a na druhé něco chybí. Na obr. 1c, 1d je pak znázorněno přesunutí tří bloků o kapacitě 8×8 znaků z pravé části mapy stínítka do spodní části mapy paměti RAM. Podobně bychom postupovali při přemísťování tří obrázků z obdélníkové nástěnky na stejně velkou čtvercovou. Tento úkon však musíme umět realizovat i logickými obvody. Na obr. 2a je celý problém znázorněn blokově: binární čítač o délce 5 bitů adresuje řádky. Výstupy čítače na adresové vstupy paměti RAM musíme překódovat.



Obr. 1. Adresace paměti RAM a stínítka

Podíváme-li se zpět na obr. 1, vidíme, že část A zůstává nezměněna a není nutné něco překódovávat. Jakmile však čítač sloupců dosáhne stavu, kdy S32 = 1, je nutné něco udělat. Blok B je charakterizován tím, že platí Ř16 = 0 a Ř8 = 0 a blok B po přesunutí zase tím, že A4 = 0 a A3 = 0. Navíc místo paměti, kam bloky přesouváme, je určeno tím, že má A8 = A9 = 1. Stejné vztahy platí i v blocích C a D. Můžeme proto napsat tabulku, kterou musí splnit překódovací obvod z obr. 2a; tabulka je na témže obrázku (obr. 2b). První sloupec vyjadřuje vztah vstupů S a Ř k výstupům A do doby, kdy platí S32 = 0 (část A mapy). Druhý sloupec pak vyjadřuje vztahy vstupů a výstupů při S32 = 1. Jak je vidět, přebytečný bit S32 vůbec překódovacím obvodem nemusí procházet, ale pouze přepíná adresy A3, A4, A8 a A9. Na obr. 2c je pro představu tato tabulka znázorněna reléovou logikou. Toto řešení by však bylo pro alfanumerický displej pomalé a proto jsem použil multiplexer 74157, který je rychlý.



Obr. 2. Překódovací obvod

Jak je vidět na obr. 1c, není využita celá paměť RAM – zbývá 64 buněk. I tak je však využití kapacity paměti dostatečné. Výhodou navrženého řešení je, že se paměť jeví, jako by měla kapacitu 2K a programátor může „sestavovat" adresu polohy znaku z čísla v řádce a z čísla řádky. Na závěr je nutné podotknout, že čísla 40 až 63 a 24 až 31 jsou zakázána a programátor na to musí pamatovat.

Další úvahy

Jak všichni dobře víte, trvá jeden televizní řádek 64 μs. Televizní signál je definován pro 625 linek, které se kreslí nadvakrát, s prokládáním. U displejů pracujících s televizním přijímačem se prokládání nepoužívá. Rozlišovací schopnost 312 linek na snímek je pro kreslení znaků dostatečná a odpadnou tak komplikace s lichým a sudým půlsnímkem. Díky tomu, že aktivní část linky, po níž se kreslí znaky, je právě 40 μs, měl jsem návrh časové základny displeje značně usnadněn. Binární čítač, který má délku 6 bitů, má celkem 64 stavů a současně může adresovat i paměť RAM v rozsahu 0 až 39. Tímto řešením se odstraní nutnost mít zvlášť čítač pro získání intervalu řádkové synchronizace a zvlášť pro adresaci RAM. Je jenom nutné zvolit vhodný okamžik, asi okolo stavu 48, kdy se bude generovat synchronizační impuls HS. Stačí tedy zvolit vstupní takt čítače 1 μs a z něho odvodit signály dalších kmitočtů.

U televizního displeje se každý znak kreslí pomocí matice bodů. Nejobvyklejší formát je 5×7 bodů a pro ten je také určen obvod MHB2500. Obvod MHB2500 je vlastně paměť ROM, která po zadání kódu znaku (6 bitů) a čísla linky znaku (3 bity) vydá na výstupu 5 bitů, určujících, má-li bod svítit nebo ne. Aby znaky nesplývaly, je nutné, aby za každou pětici bodů byla jedna „černá" mezera. Tím byl dán další parametr a to kmitočet, kterým se řídí jas paprsku podle obsahu obvodu MHB2500; kmitočet se jmenuje bodový (dot frequency). Tím byl stanoven kmitočet oscilátoru 6 MHz. Od tohoto kmitočtu se budou odvozovat další časy v displeji.

Další částí časové základny displeje je čítač linek pro jeden znakový řádek. Obvod MHB2500 má 7 linek pro znak a jednu prázdnou pro mezeru. Oddělení znakových řádků pouze jednou mezerou nedává dobře čitelný text a navíc by neumožnilo zobrazení tzv. kurzoru. Kurzor umožňuje programátorovi, aby určil místo, kam se bude psát nový znak nebo místo pro změnu znaku atd. Nejčastěji se kurzor zobrazuje jako podtržení znaku. Proto jsem volil 10 linek na jeden znakový řádek. Linky 0 až 6 kreslí znak, linka 7 je vždy zatemněná, linka 8 je vyhrazena pro kurzor a linka 9 je také vždy zatemněna. Čítač linek je tvořen obvodem 7490, který dělí deseti. Za čítačem linek je pak zařazen čítač znakových řádků, který adresuje pole dalších 40 znaků v paměti RAM. Vrátíme-li se zpět k obr. 2 vidíme, že čítač řádku má délku 5 bitů a má tudíž 32 stavů, z nichž stavy 0 až 23 adresují paměť RAM. Zařazením čítače linek a čítačů řádků za sebe získáme čítač modulo 320 a to je číslo velice blízké počtu televizních linek, jichž je 312. Snímková perioda pak bude 20,48 ms a bude se lišit od normované pouze o 2,5 %, a to pro synchronizační obvody TV přijímače zcela vyhovuje. Jak je vidět, zařazením čítačů MOD 6, MOD 64, MOD 10 a MOD 32 je možno realizovat najednou časovou základnu pro vytvoření period řádkové i snímkové synchronizace i čítače pro adresaci paměti RAM. Na obr. 3 je časová základna AND-1, navržená podle předešlých úvah. Je znázorněn již i posuvný registr, který převádí paralelní výstup generátorů znaků ROM na obrazový signál (video), který moduluje jas obrazovky.

Kód zobrazovaných znaků

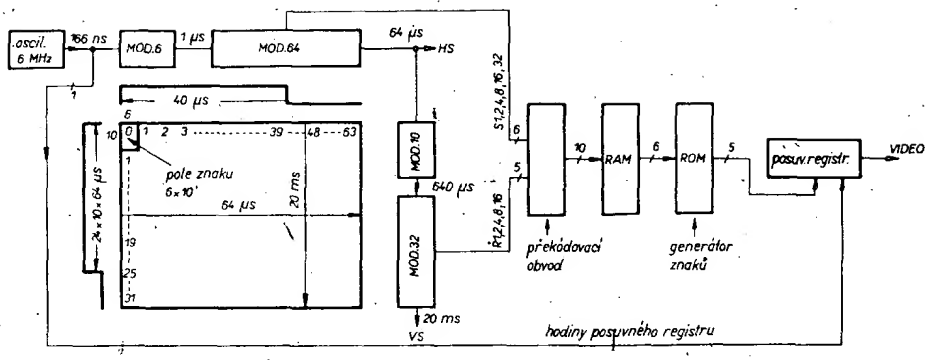
Generátor znaků má v sobě matici 5×7 bitů pro 64 znaků. Kód znaků odpovídá 6bitovému kódu ASCII pro skupinu grafických symbolů. Mikropočítač má však délku slova 8 bitů a proto je škoda nejvyšší dva bity D7, D6 nevyužít. Naskýtá se velké množství možností, neboť zbývající tři kombinace těchto bitů jsou volné a kombinace 00 bude kreslit znaky přímo tak, jak jsou zakódovány v obvodu MHB2500. Jedna kombinace musí být využita pro zobrazení znaku s kurzorem a další dvě zbývají.

Při návrhu displeje AND-1 jsem vycházel z toho, že to má být displej alfanumerický, a že grafický displej bude vyvinut později. Proto jsem nepoužil tzv. pseudografiku, při níž právě bit D7 přepíná na zobrazení pseudografických znaků, skládajících se ze šesti čtverečků (TRS 80). Protože systém JPR-1 je určen pro skutečné aplikace a ne pro hry, zvolil jsem zbývající dvě kombinace bitů tak, aby bylo možno odlišit na obrazovce důležité, méně důležité a běžné zprávy.

Podle bitů D7 a D6 kreslí displej AND-1 podle obsahu ROM takto:

bity D7 a D6 = 0 – běžný znak 5×7,
bity D7 = 0 a D6 = 1 – běžný znak, který bliká,
bity D7 = 1 a D6 = 0 – běžný znak, ale bliká linka 8, která představuje kurzor,
bity D7 = 1 a D6 = 1 – znak s dvojitou šířkou (10 × 7).

Návrh displeje vychází z toho, aby uvedené „parády" nevyžadovaly moc obvodů a byly přesto už itečné. Dvojitá šířka znaku je běžná u tiskáren a je výhodná, má-li se na obrazovce zobrazit nějaké číslo nebo údaj, který musíme číst z větší dálky (příklad: zobrazení souřadnic X, Y u mikropočítačem řízené vrtačky).



Obr. 3. Znázornění časové základny AND-1

Blikající znak vyšel prakticky zadarmo, protože jsem pro dekodér bitů D7 a D6 použil paměť PROM a ta má zaveden kmitočet blikání pro zajištění funkce blikajícího kurzoru.

Středění textu na stínítku

To, že je text často příliš na kraji nebo nahoře, zjistilo již hodně uživatelů osobních mikropočítačů. Proto jsem obvody, které jsem ušetřil na časové základně, věnoval na možnost nastavit polohu textu na stínítku. Jedná se vlastně o to, zajistit přítomnost synchronizačních impulsů HS a VS ve vhodný okamžik.

Ten, kdo si prohlédl pozorně obr. 3, tuší, že impuls HS musí vznikat asi při stavu 48 čítačů MOD 64 a impuls VS asi při stavu 25 čítače MOD 32 a ne na konci, jak je nakresleno. U prvního vzorku AND-1 byly tyto stavy dekódovány dekodéry 7442 nebo 3205. Tento způsob je obvyklý, ale špatně se realizuje jemné přepínání dekodéru, který při posouvání textu musí dekódovat stavy 47, 48, 49 apod. podle přání uživatele.

Chceme-li k zobrazení použít i videomonitor, buď zahraniční nebo AZJ 462 z TESLA Orava, zjistíme navíc, že monitory nemají s TV přijímačem mnoho společného. Generátory signálů pilovitého průběhu jsou navrhovány vzhledem k velkým nárokům na linearitu vychylování a proto monitory nemají volně běžící rozkladové obvody jako TV přijímače.

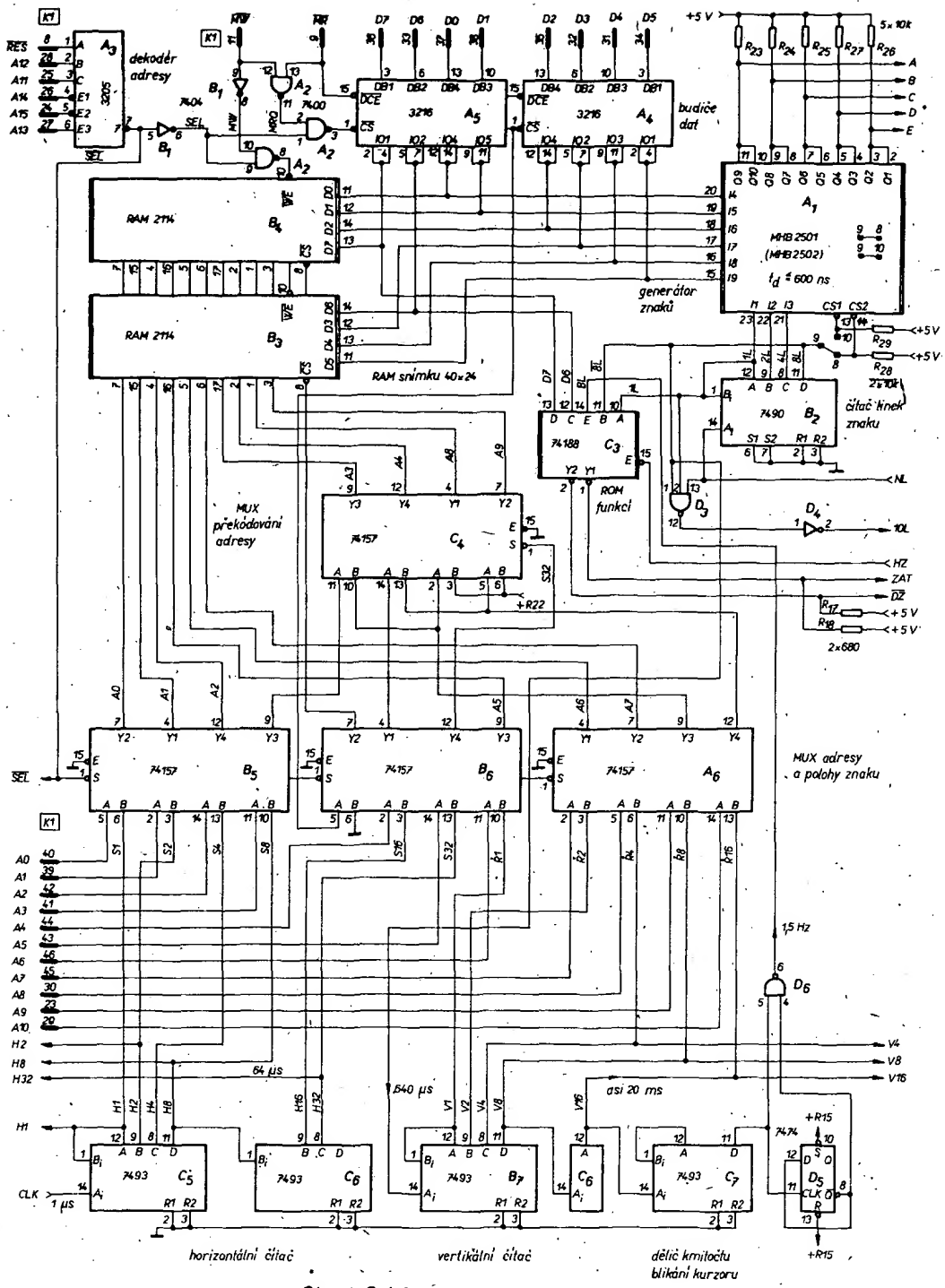
I když však mají HS a VS správnou šířku i periodu, nemáme ještě vyhráno. Některé monitory začínají kreslit hned po skončení synchronizačního impulsu, některé dříve apod. Nejsem odborníkem v této technice, proto jsem chtěl pouze upozornit, že poloha signálů HS a VS je u monitorů jiná než v TV signálu. Zejména proto jsem zvolil složitější nastavování okamžiku HS a VS, než je obvyklé – okamžik vyslání synchronizačních impulsů lze tedy posouvat a změnou konstant *RC* u monostabilních obvodů lze měnit i šířku impulsů.

Generátory signálů pilovitého průběhu v monitorech pro výpočetní techniku (tyto monitory jsou jiné než videomonitory pro videotechniku), dodávají průběhy podle vstupních signálů HS a VS. Nejsou-li tyto signály připojeny, signály pilovitého prů-

Tab. 1. Zapojení konektoru K1 desky AND-1

Č.	Signál	Název	Typ	Č.	Signál	Název	Typ
7				8	$\overline{RES}$	nulování	INP
9	$\overline{MR}$	čtení z paměti	INP	10			
11	$\overline{MW}$	zápis do paměti	INP	12			
13				14			
15	+5 V	napájení	NAP	16	+5 V	napájení	NAP
17	+5 V		NAP	18	+5 V		NAP
19	0 V	zem	NAP	20	0 V	zem	NAP
21				22			
23	A9		INP	24	A15		INP
25	A11	adresa	INP	26	A14	adresa	INP
27	A13		INP	28	A12		INP
29	A10		INP	30	A8		INP
31	D4		BD	32	D3		BD
33	D6	data	BD	34	D5	data	BD
35	D2		BD	36	D7		BD
37	D0		BD	38	D1		BD
39	A1		INP	40	A0		INP
41	A3	adresa	INP	42	A2	adresa	INP
43	A5		INP	44	A4		INP
45	A7		INP	46	A6		INP
53	0 V	zem	NAP	54	0 V	zem	NAP
55				56			
57	(−12 V)	není nutné	NAP	58	(−12 V)		NAP
59	(INT1)	není nutné	OUT	60			

Číslo konektoru:	K1	Konektor:	TY 517 62 11	INP – vstup
Deska/zařízení:	AND-1	Protikus:	TX 518 62 12	BD – obousměrný
Klíčování:	C6			OUT – výstup
				NAP – napájení



Obr. 4. Schéma desky AND-1, list 1

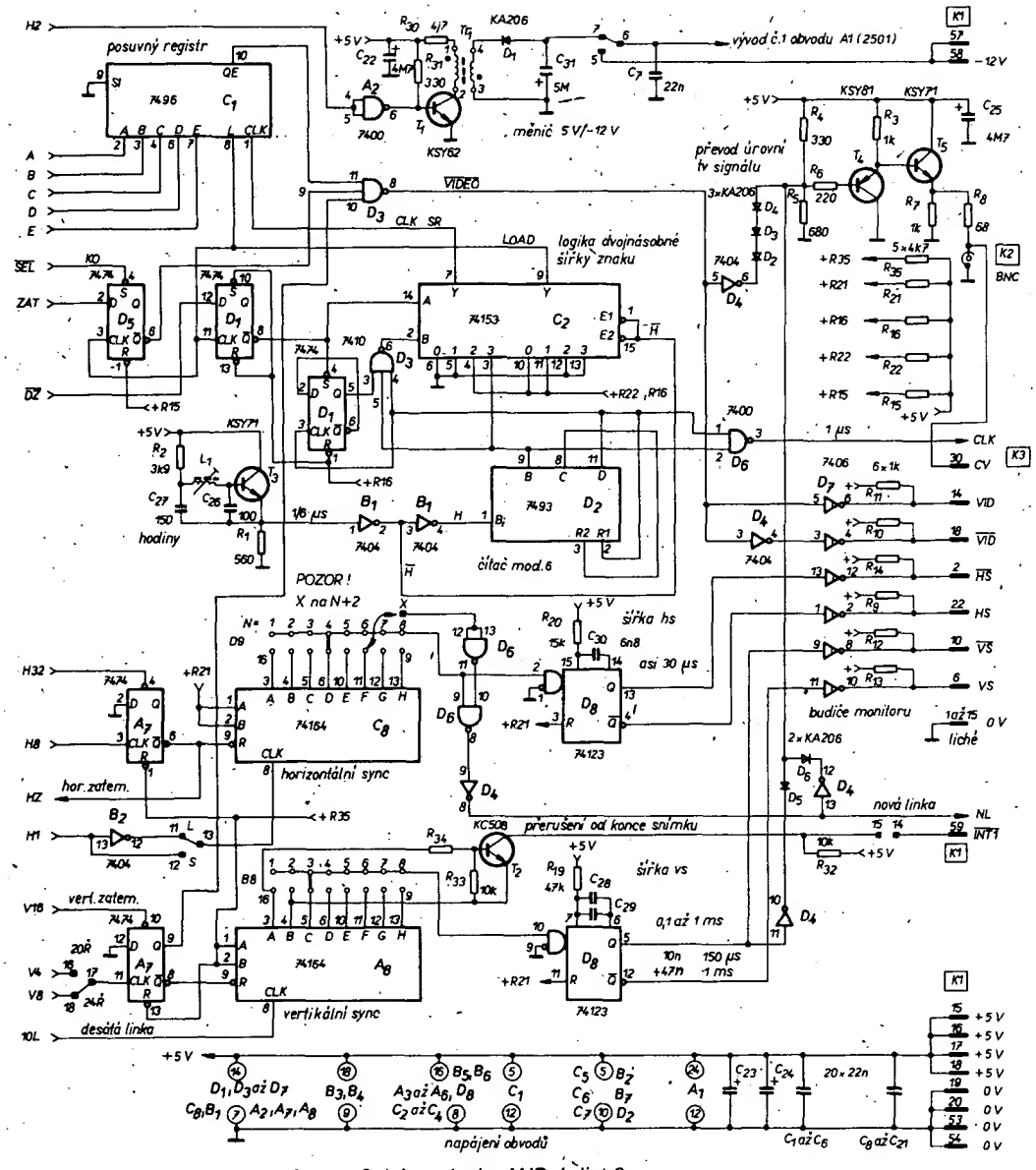
běhu nevznikají a obvykle nejde ani vysoké napětí. Je proto nutné prostudovat pozorně požadavky na vstupní signály u každého monitoru. Může se dokonce stát, že při nesprávných šířkách nebo periodách vstupních signálů se zničí výkonové tranzistory v monitoru. Vstupní vazba signálu HS bývá transformátorová a při krátkém signálu HS pracuje pak tranzistor v aktivní oblasti a přehřeje se.

Popis zapojení AND-1

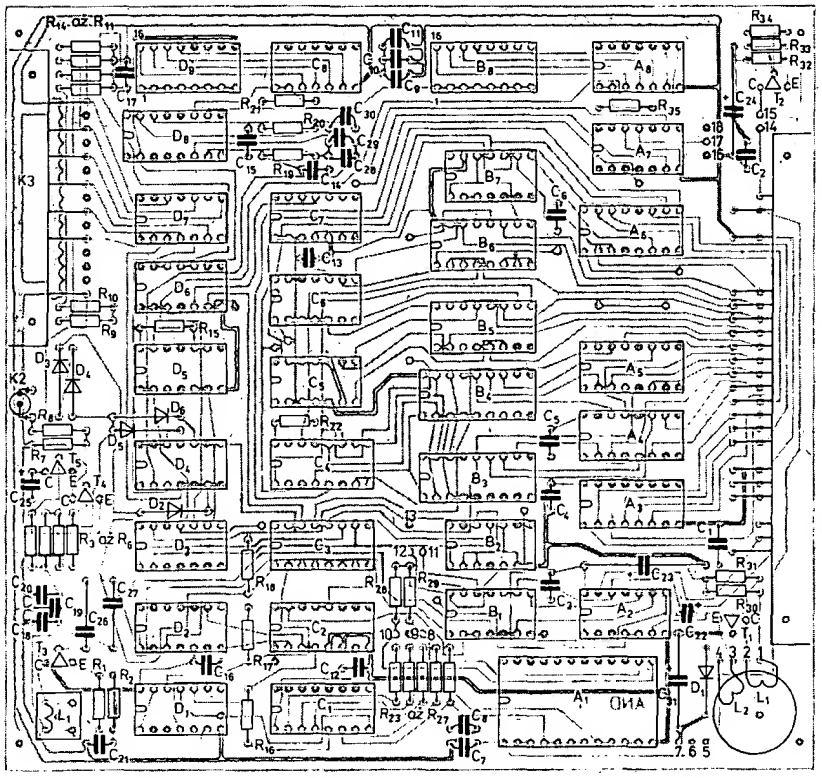
Na obr. 4 a 5 jsou schémata desky AND-1. Na obr. 6 je rozložení součástek na desce s plošnými spoji. Pro oživení a opravy je nutné znát funkci jednotlivých obvodů a proto sledujte text pečlivě. Něco už víme z předchozích úvah, ale to pro znalost funkce rozhodně nestačí.

1. Z hlediska spolupráce displeje s procesorem JPR-1 je důležité zapojení konektoru K1, který je připojen ke sběrnici ARB-1 (tab. 1). Procesor komunikuje s displejem jako s pamětí o kapacitě 2K byte, která leží v rozsahu adres 3800 až 3FFF. Pro komunikaci používá procesor signály $\overline{MR}$ a $\overline{MW}$, to znamená, že může z paměti číst i do ní zapisovat. Je proto možné zajistit programem „rolování" řádek a jiné operace. Takto organizované paměti displeje říkáme VIDEORAM. Paměť by bylo možno použít i jako běžný RAM procesoru, musí se však dát pozor na to, že nemá 2K byte, ale jen 40×24 byte. Možnost přístupu do paměti ze sběrnice i z vnitřních obvodů displeje (dual port RAM) zajišťuje multiplexer adresy, tvořený obvody B_5, B_6 a A_6. Adresa se přepíná signálem SEL (vývod 1 obvodu $B_5 = 1/B_5$), který je generován dekodérem adresy A_3. V okamžiku výběru desky jsou od paměti RAM odpojeny čítače a jsou připojeny adresové linky A0 až A10. Toto přepnutí trvá prakticky jen jeden strojový cyklus mikroprocesoru a po tuto dobu je stínítko zatemněno. Ten, kdo nechce mít na stínítku černé čárky vzniklé zápisem „za chodu", může použít signál $\overline{INT}$ 1 a přerušením manipulovat s obsahem VIDEORAM jen při zpětném běhu snímku. Na přenos dat je pak k dispozici asi 5 až 7 ms (podle počtu zobrazených řádků, 24 nebo 20).

2. Přenos dat po sběrnici je obousměrný. Přenos umožňují obousměrné budiče dat A_4, A_5. Jejich řídicí signály přicházejí z hradel A_2 a B_1. Proto, aby byly tyto budiče aktivní v jednom či druhém směru, musí být přítomen signál MRQ (požadavek na paměť) a současně musí



Obr. 5. Schéma desky AND-1, list 2



Obr. 6. Rozložení součástek na desce AND-1

být na sběrnici správná adresa desky (signál SEL = výběr). O směru přenosu pak rozhodne signál $\overline{MR}$ (15/A_5 a 15/A_4). Je-li deska vybrána a je-li zápis ($\overline{MW}$), pak vzniká i signál $\overline{WE}$ (10/B_4) a data se píší do VIDEORAM B_3 a B_4. Signál RESET, přicházející ze sběrnice, zavírá dekodér A_3 a brání tak „konfliktům" třístavových obvodů sběrnice a desky.

3. Obvod C_4 je již známý překódovací obvod, umožňující „nebinární" formát dat na stínítku při zachování binární adresace.

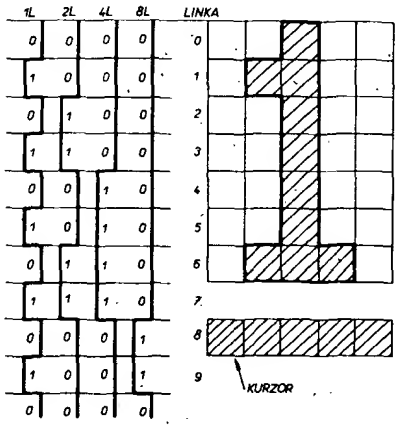
4. Čítače C_5 a C_6 tvoří čítač MOD 64 a jeho výstupy jsou značeny jako H1 až H32 a současně S1 až S32. Toto dvojí označení odpovídá dvojí funkci čítače. Signály H slouží ke generaci horizontálních synchronizačních a zatmívacích impulsů, signály S k výběru sloupců znaků z paměti. Čítač B_7 a zbytek C_6 tvoří čítač MOD 32 a také má dvojí funkci. Tento čítač je inkrementován signálem 8L, neboli vždy po desáté lince TV rastru.

5. Čítač tvořený obvody C_7 a D_5 slouží k dalšímu dělení kmitočtu snímkového signálu až na 1,5 Hz; zhotovit by ho bylo

možno i z časovače LM555, tento obvod se však bohužel u nás nevyrábí. Hradlo D_6 dodává signál se střídou 1:3 pro zatmívání, aby byla doba zatmění kratší než doba svícení.

6. Čítač B_2 je jedním z nejdůležitějších obvodů desky. Je to vlastně čítač MOD 10, který počítá TV linky. Jeho obsah je inkrementován signálem NL – nová linka vždy každých 64 μs. Hradlo D_3 vydá vždy při každé desáté lince signál 10L, který slouží obvodům snímkové synchronizace jako hodiny pro zpožďovací obvod signálu VS.

Výstupní signály 1L a 8L jdou do paměti PROM, kde se z nich dekóduje poloha řádku, v němž je kurzor (řádka 8). Signály 1L, 2L a 4L říkají generátoru znaků (A_1), jaká linka znaku se právě kreslí. Průběhy na čítači B_2 jsou na obr. 7.



Obr. 7. Průběhy čítače B2

7. Generátor znaků A_1 je zasunut do objímky a je ho možné zaměnit za obvod MHB2502, který „umí" azbuku. V tom případě je nutné přepojit spojku 9–8 na 9–10. Do generátoru znaků jdou ještě „vnitřní" data z paměti RAM. Po krátký okamžik zápisu do RAM jsou na vstupu generátoru znaků zapisována data. Při práci displeje je obrazovka při signálu SEL zatemněna, takže uvedená skutečnost nevadí. Výstupy generátoru znaků vedou na vstupy posuvného registru C_1 a jsou „přitaženy" odpory na +5V, protože se k vytvoření mezery mezi řádky používá zavření výstupů signálem 8L na vstupy CS1 a CS2. Je to vlastně trik, protože tím se řádky 8, 9 a 10 rozsvítí a my je potom můžeme ovládat zatmíváním z paměti PROM (C_3). Tak je zajištěno svícení kurzoru v deváté lince.

8. „Hodiny" celé desky jsou odvozeny z oscilátoru 6 MHz, převzatého z TV her. Zatím si řekněme, že vedou do čítače MOD 6 (D_2) a výstup 3 hradla D_6 generuje hodinový signál s periodou 1 μs pro horizontální čítač. Hradlo D_6 vybírá okamžik přičtení do čítače o něco dříve než čítač přeteče, dekóduje totiž stav 5 a čítač C_5 se inkrementuje na začátku stavu 5, neboli o 166 ns dříve, než by měl! Toto řešení je nutné proto, že celý systém (čítač, multiplexery RAM a ROM) má zpoždění průchodu signálu na hranici 1 μs a to je právě takt kreslení znaku. Proto je nutné mít rychlou „verzi" 2501/2 i RAM. Předčasné zvětšení obsahu čítače tak zmenší nároky na rychlost obvodů. Ideálními čítači pro displej obvody 7490A a 93A rozhodně nejsou, ale jiné bohužel nemáme (obvody 74193 mají velký odběr a jsou dražší). Na čítačích se vytváří, dá se říci, zbytečná část celého zpoždění průchodu signálu. Paměti zpoždění mít musí! Doporučuji však vybrat rychlé obvody 2114 a ty pomalejší použít v procesoru.

9. Systém synchronizace je pro H i V stejný. Klopný obvod A_7 se odblokuje po dosažení stavu 32H čítače. O 8 taktů později, to je ve stavu H = 40, se klopný obvod $A_7/6$ „nahodí". Aby nemusel být na vstupu D odpor, používám obvod 7474 vlastně v negativní logice. „Nahození" obvodu odblokuje posuvný registr C_8, který čítá impulsy H1 s periodou 2 μs. Připojováním vstupu $2/D_8$ na různé výstupy posuvného registru tak získáme zpoždění po krocích 2 μs. Kdyby někdo chtěl posouvat polohu znaků jemněji, musí si přepojit spojku 11–13 na 12–13. Tato spojka určuje, začíná-li synchronizace v lichém nebo sudém stavu čítače H (nastavuje se nejprve hrubě spojkami v pozici obvodu D_9 a pak jemně spojkou lichá/sudá). Hradlo D_6 vyrábí impuls pro synchronizaci TV přijímače. Odpovídající šířka impulsu je 4 μs a protože má registr C_8 takt 2 μs, stačí připojit bod *X* na výstup N + 2, než spojka a hradla D_6 a D_4 vygenerují přes diodu D_6 správný impuls. Není pak nutné nastavovat přesně časovou konstantu obvodu D_8 pro HS. Monostabilní obvod pro šířku impulsu HS je vlastně určen pouze pro monitor. Monitor AZJ 462 vyžaduje šířku HS 30 μs.

10. Funkce vertikální synchronizace začíná dosažením stavu V16, kdy se odblokuje klopný obvod A_7. Obvod je pak nahozen stavem V = 20 (spojka 16–17) nebo V = 24 (spojka 18–17). Spojka je vlastně určena pouze pro monitor, u něhož lze nakreslit 24 znakových řádek pohodlně. Zpoždění je nastaveno v krocích 0,64 ms a impuls VS pro TV přijímač (100 μs) i monitor (AZJ 462 = 1 ms) vytváří obvod D_8. Proto má tento obvod také dva kondenzátory C_{28} a C_{29}, aby je nebylo nutné měnit. Impuls pro VS TV přijímače je veden přes diodu D_5 do převodníku úrovní. Na první dva výstupy posuvného registru je připojen tranzistor T_2, pracující ve funkci derivačního obvodu náběhové hrany impulsu. Po nahození $8/A_7$ dá T_2 žádost o přerušení INT 1. Funkci přerušení je možné zařadit spojkou 15–14.

11. Důležitá je funkce zatmívání. Videosignál nesmí procházet ven trvale. Kreslí-li se šestý bod linky znaku, je zatmění zajištěno uzemněním vstupu SI posuvného registru C_1. Po dokreslení řádku 20 nebo 24 se obrazovka zatmívá spojením výstupu $9/A_7$ a $10/D_3$. Při kreslení linek znaků se zatmívá poněkud složitěji klopným obvodem $D_5/6$. Tento klopný obvod má takt synchronní s naplňováním posuvného registru C_1 – signál LOAD. Kdy se má zatmívat, určuje výstup 1 paměti PROM C_3: jednak při signálu HZ (stav čítače H větší než (40), protože ten zavře celou PROM, jednak podle obsahu bitu Y1 naprogramovaného do C_3. Zatmívá se v linkách 7, 8, 9 znaků bez blikání, v linkách 7 a 9 při kurzoru, v lince 8 při kurzoru, je-li BL = 0, a při všech linkách blikajícího znaku, je-li BL = 0. Typ znaku se určí z kombinace bitů D7 a D6 a číslo linky ze vstupů 1L a 8L.

12. Obvod, který generuje signály LOAD a CLK, umí také zařídit dvojnásobnou šířku znaku. To, že se jedná o znak s kódem $D_7 = D_6 = 1$, pozná paměť PROM C_3 a vydá signál DZ. Potom se nastaví výstup $8/D_1$ na H a odblokuje se dělič, tvořený druhou polovinou obvodu D_1. Současně se změní úroveň vstupu A multiplexeru C_2. Na výstupu $7/C_2$ se objeví signál poloviního kmitočtu hodin (z výstupu B obvodu D_2). Hodinový signál posuvného registru C_1 (CLK SR) a signál paralelního naplnění registru (LOAD) mají průběh podle obr. 6. Jak je vidět, je vždy 5 impulsů posuvu a jeden pro naplnění. Vznik těchto signálů je řízen stavem čítače D_2 (má 6 stavů, 0 až 5). Jakmile je na $8/D_1$ úroveň H, zapojí se do hry i dělič (výstup $5/D_1$) a vznikají „poloviční hodiny" CLK, vždy pouze v lichém stavu čítače D_2 (řízené výstupem B čítače). Navíc dělič $6/D_1$ zabrání průchodu signálu LOAD, je-li na $5/D_1$ úroveň L. Tím dostaneme také jen každý druhý LOAD. Celý logický obvod je možno realizovat hradly NAND. Použití multiplexeru 74153 je však elegantní a zjednoduší i desku s plošnými spoji.

13. Zbývající obvody na desce jsou již jednoduché. Tranzistor T_1 vyrábí napětí −12 V pro napájení generátoru znaků MHB2500. Má-li systém −12 V, je možno neosadit součástky měniče a přepojit spojku 6–7 na 6–5.

Tranzistory T_4 a T_5 pracují společně s diodami D_2 až D_6 jako převodník úrovní TTL na videosignál TV přijímače.

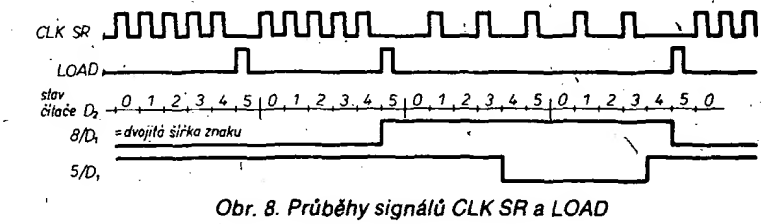
Výkonová hradla s otevřeným kolektorem 7406 (obvod D_7) zesilují výstupní signály, protože vstupy monitoru jsou zakončeny tvrdými děliči 180/220 Ω. Odpory na výstupech D_7 slouží tedy k měření průběhů a nejsou nutné. Konektor K3 je určen pro připojení monitoru AZJ 462. Volné špičky K3 je možno využít pro „vytažení" dalších signálů, děláme-li s deskou nějaké pokusy, je např. možné vytáhnout externí hodiny H a neosadit tranzistor T_3. Pak lze pomocí fázového závěsu CMOS 4046 synchronizovat snímkový kmitočet se síťovým kmitočtem apod.

Zhotovení a oživení desky AND-1

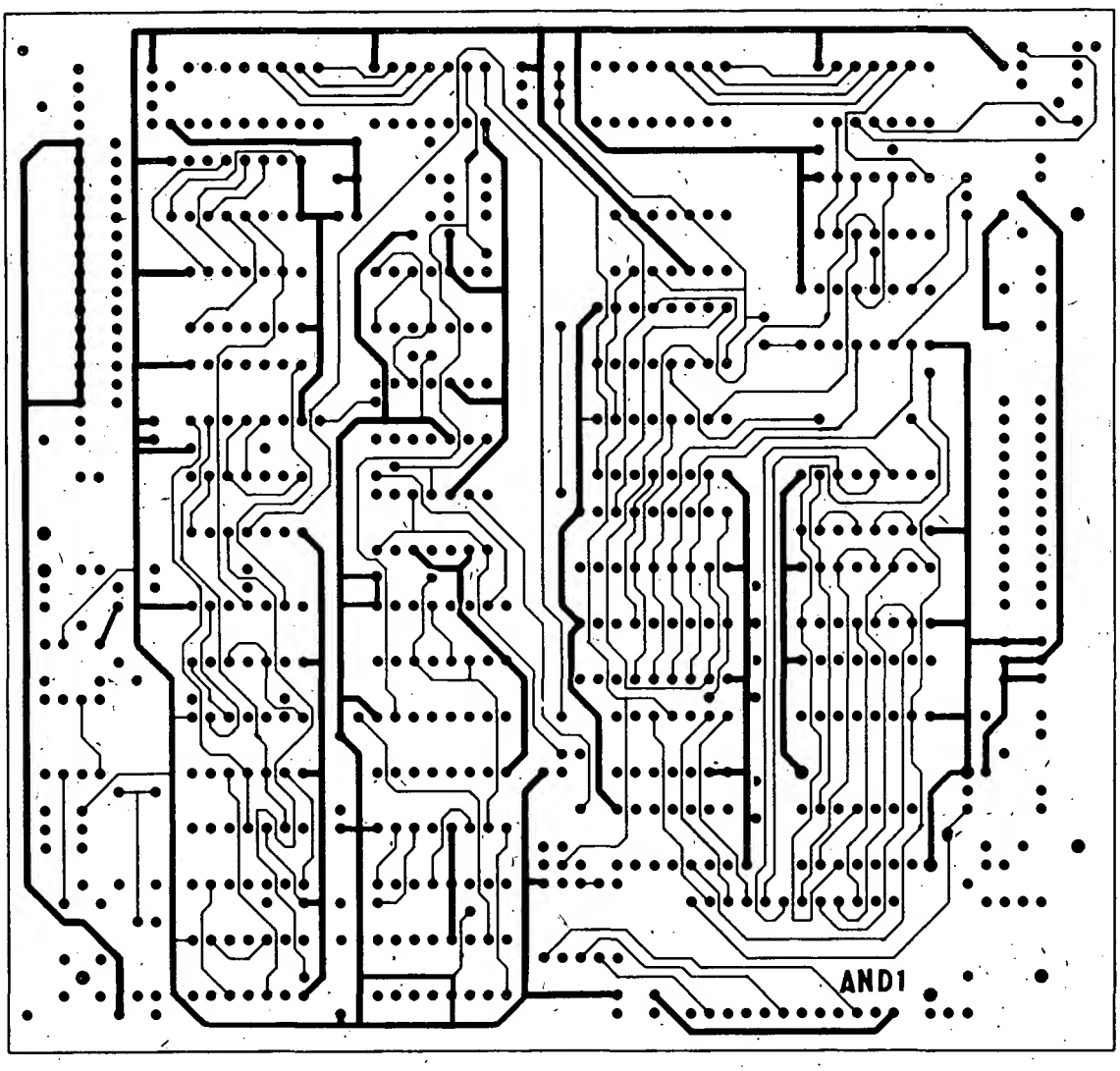
Máme-li desku s plošnými spoji (obr. 9 a 10) s prokovenými děrami, je osazení desky AND-1 jednoduché. Máme-li desku bez prokovených děr, musíme nejprve zajistit průchod spojů z jedné strany na druhou tam, kde budou objímky. O tom, jak se to dělá, jsem se již zmínil v kapitole o práci s mikroprocesory. Objímky jsou u obvodů A_1, C_3, B_4 a B_3. Objímky pro paměti 2114 (18 vývodů) musíme zhotovit ze dvou našich objímek pro IO. Máme-li obvody, které patří do objímek, vyzkoušeny v jiném vzorku desky AND-1, pak nemusíme objímky používat.

Potom navineme cívku L_1 a transformátor Tr_1 (drát o Ø 0,2 mm, nejlépe se samopájitelnou izolací).

Umístění propojek, které volí různé funkce, závisí na tom, budeme-li chtít funkce měnit, nebo je-li deska určena pro pevnou aplikaci. Velký počet přepínacích



Obr. 8. Průběhy signálů CLK SR a LOAD



Obr. 9. Plošný spoj desky AND-1 strana součástek

propojek zhoršuje spolehlivost, a proto je lepší pájet propojky napevno. Jinak lze propojovací spojky zhotovit z drátu a jako špičky použít dutinky z konektorů FRB. Protože jsou pak dutinky blízko sebe, je lépe na prostřední navléknout bužírku. V pozicích B_8 a D_9 je počítáno s místem pro přepínače v pouzdře DIL, ale postačí objímka pro IO (16 vývodů) nebo propojka drátkem.

Před osazením konektorů z nich vyjmeme nepoužité špičky. Osazení a zapájení součástek by nemělo činit potíže. Po skončení pájení pečlivě prohlédneme celou desku, nejlépe lupou. Zkontrolujeme propojky a můžeme začít s oživováním. Používáme-li objímky, počkáme se zasunutím obvodů, dokud nepřekontrolujeme napájení na objímkách. Měnič na –12 V je navržen tak, že dává správné napětí pouze při typickém odběru obvodu MHB2500.

Pro oživování desky použijeme přípravek TST-03. Sepnutím přepínače ENABLE zajistíme připojení třístavových budičů adresových a řídicích signálů na testovací konektor K1. Do přípravku zasuneme desku AND-1. Nejprve proměříme funkci signálů, které jdou na sběrnici. Přepínáme jednotlivé spínače adres a kontrolujeme sondou, jdou-li na vstupy obvodů B_5, B_6, A_6 a A_3 správně adresy. Potom zkontrolujeme správnou funkci dekodéru adres A_3.

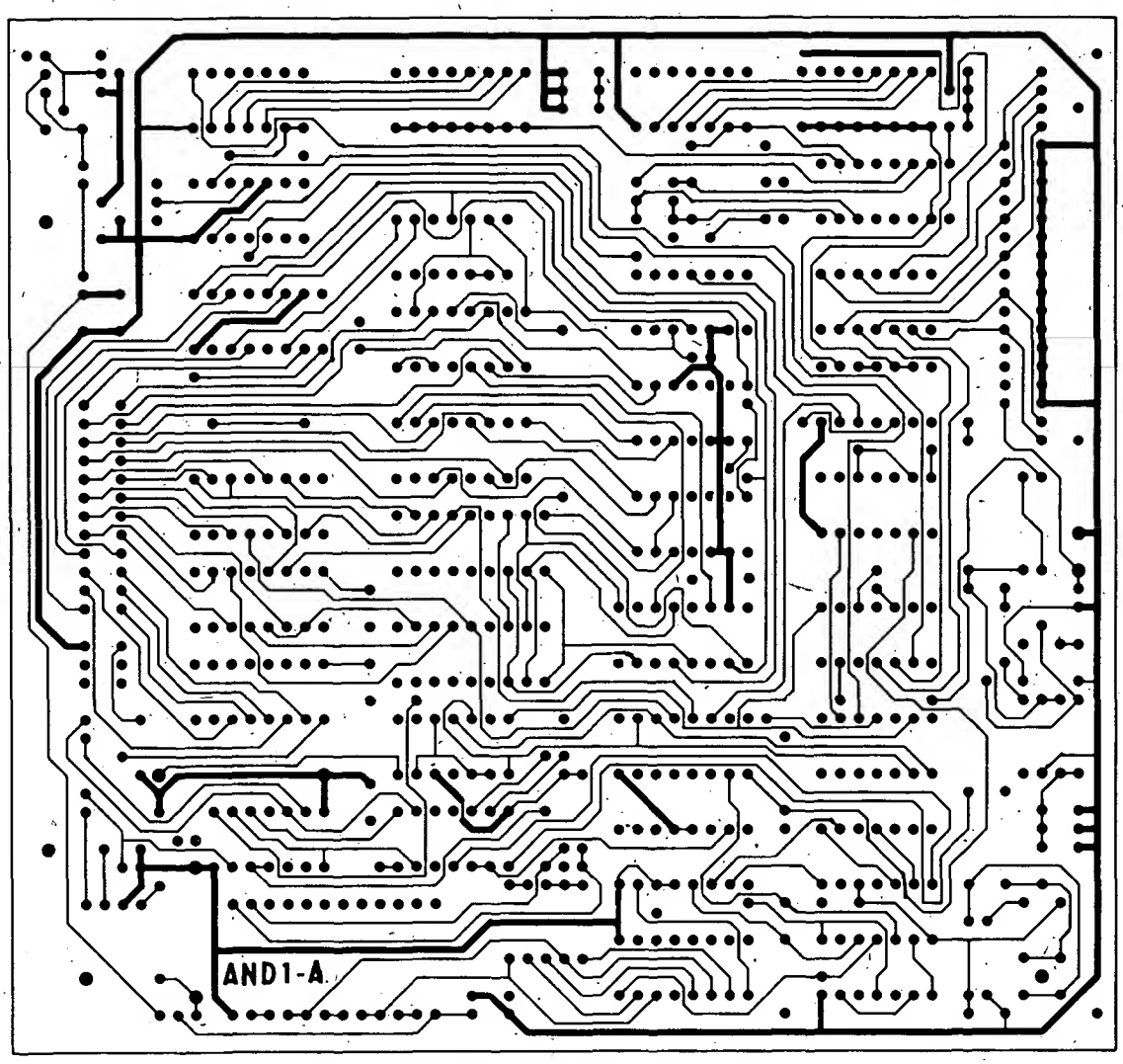
Tab. 2. Zapojení konektoru K3 desky AND-1

Č.	Signál	Název	Typ	Č.	Signál	Název	Typ
1	0 V			2	$\overline{HS}$	hor. sync.	OKB
3	0 V			4			
5	0 V			6	VS	vert. sync.	OKB
7	0 V			8			
9	0 V			10	$\overline{VS}$	vert. sync.	OKB
11	0 V			12			
13	0 V			14	VID	video + zat.	OKB
15	0 V	zem		16			
17	0 V			18	$\overline{VID}$	video + zat.	OKB
19	0 V			20			
21	0 V			22	HS	hor. sync.	OKB
23	0 V			24			
25	0 V			26			
27	0 V			28			
29	0 V			30	CV	video + sync.	TV sig.

Číslo konektoru:	K3	Konektor:	TY 513 30 11	OKB otevřený kolektor, výkonový
Deska/zařízení:	AND-1	Protikus:	TX 514 30 13	
Klíčování:	A-4			

Signál SEL má mít úroveň L při A11 = A12 = A13 = H a A14 = A15 = L. Je-li SEL, můžeme překontrolovat, procházejí-li adresy až na paměť RAM (B_3 a B_4). Potom zkontrolujeme čtení a zápis do paměti. Tlačítky MW a MR na přípravku zkusíme zapsat a přečíst třeba samé nuly (L) a samé jedničky (H). Nejde-li to, překontrolujeme obvody A_2, B_1, A_4 a A_5 logickou sondou.

Dále proměříme funkci časové základny čítačem nebo osciloskopem. Začneme od generátoru hodin, jehož kmitočet má být 6 MHz. Signál CLK má mít periodu 1 µs. Průběhy signálů CLK SR a LOAD zatím neměříme, protože po zapnutí má paměť RAM libovolný obsah a průběhy těchto signálů nejsou periodické (střídají se běžné a dvojité znaky). Proměříme celou časovou základnu podle poznámek ve schématu. Pracuje-li celá časová základna správně, je perioda signálu HS = 64 µs a signálu VS asi 20 ms. Podle toho, použijeme-li televizor nebo jednotku AZJ 462, nastavíme časy monostabilního obvodu D_8. Pro televizor není průběh na 13/D_8 zajímavý, neboť špička HS se odvozuje od výstupů posuvného registru C_8 (špička X musí být připojena ob jednu špičku za propojku, která určuje začátek HS) a HS = 4 µs. Druhý monostabilní obvod má mít šířku impulsu 100 až 150 µs (C_{28} = 10 nF). Pro AZJ 462 má mít první monostabilní obvod periodu asi 30 µs



Obr. 10. Plošný spoj desky AND-1 spodní. strana

a druhý 1 ms (přidáme C_{29} = 47 nF). Potom teprve můžeme připojit televizor nebo monitor. V tab. 2 je zapojení konektoru K3, který slouží pro připojení jednotky AZJ 462 nebo TVP. Na obr. 11 je propojení AZJ 462 s deskou AND-1. Na obr. 12 je úprava TVP Satelit.

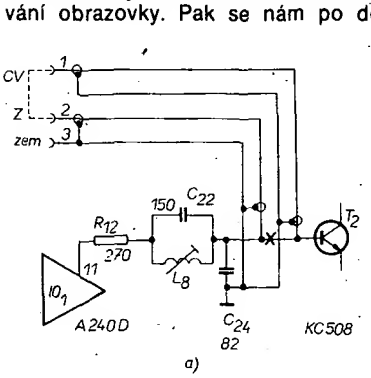
Pozor!!! Jako TVP smí být použit pouze přijímač se síťovým transformátorem, jinak je nutno použít vf modulátor jako u TV her!

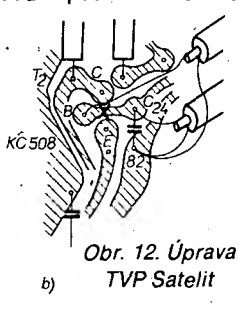
Oživujeme-li desku na přípravku TST-03, musíme si uvědomit, že signál SEL zatmívá obrazovku a chceme-li na obrazovce něco vidět, musíme zrušit SEL, třeba přepnutím A15 = 1. Můžeme také uzemnit $1/D_5$ a tím zrušit částečně zatmívání obrazovky. Pak se nám po dobu trvání SEL a MW objeví kód znaku přímo na vstupech generátoru znaků a můžeme si v klidu vyzkoušet jeho funkci.

Další oživování desky závisí na tom, co z její celkové funkce „nechodí". Jednotlivé typy znaků (normální, blikající, s kurzorem a dvojnásobný) jsou dekódovány pamětí PROM C3. Její obsah je uveden v tab. 3. Průběhy signálů CLK SR a LOAD podle obr. 8 pro jednoduchý i dvojnásobný znak se nejlépe měří, má-li obrazovka všude stejný znak. Nakonec ještě zkusíme do paměti RAM zapsat různé znaky na předem určenou polohu podle tab. 4. Po skončení oživování můžeme paměť a celý displej AND-1 testovat již systémem JPR-1. Výhodou displeje AND-1 je možnost posouvat zobrazovaný text na stínítku obrazovky a také to, že paměť RAM je vlastně pamětí procesoru, takže můžeme text snadno „rolovat".

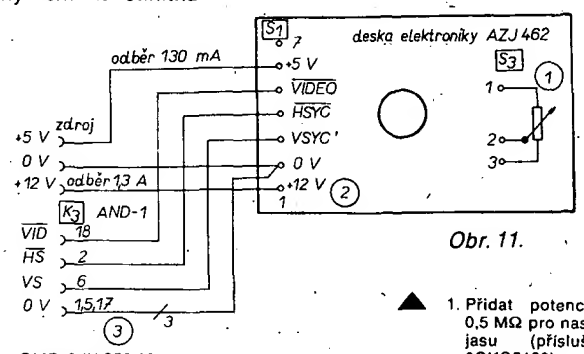
Programování AND-1

Paměť RAM displeje AND-1 má pevně zvolenou adresu a začíná od adresy 3800H. Pro rychlejší orientaci jsou v tab. 4 vypsány adresy jednotlivých znaků na stínítku v matici 40 × 24 v kódu HEX. Při použití znaků s dvojitou šířkou nesmíme zapomenout na to, že následující poloha na stínítku (v rámci jednoho řádku) se přeskočí a znak na této adrese nebude





Obr. 12. Úprava TVP Satelit



Obr. 11.

1. Odsát a odpájet jednotku OMF, 6PN 052 16.
2. Přerušit spoj C_{24} – báze T_2.
3. Připájet dva souosé kablíky podle b) a protáhnout je děrami v krytu jednotky.
4. Složit a zapájet OMF do desky s plošnými spoji.
5. Vyvést vývody na konektor, připevněný na zadní stěně TVP.
6. Zkratovat spojkou vývody 1–2 při běžném příjmu TV programů.

1. Přidat potenciometr 0,5 MΩ pro nastavení jasu (příslušenství 9QKO5109).
2. Pro S1 použít zásuvku WK 180 23.
3. Jako vodič použít „twist" VFDP90 – 2× 0,4 mm.

Tab. 3. Obsah paměti 74188 (C3) pro dekodér funkcí znaků

Řádek	Vstupy						Výstupy		Pozn.		
	EN	E	D	C	B	A	Y2	Y1	BL = 0 tma	D_7, D_6	linka
0	0	0	0	0	0	0	1	0	svítí	norm. znak	0 až 7
1	0	0	0	0	0	1	1	0			
2	0	0	0	0	1	0	1	1			8
3	0	0	0	0	1	1	1	1			9
4	0	0	0	1	0	0	1	1	tma	blik. ZN	0 až 7
5	0	0	0	1	0	1	1	1			
6	0	0	0	1	1	0	1	1			8
7	0	0	0	1	1	1	1	1			9
8	0	0	1	0	0	0	1	0	tma	blik. kurz.	0 až 7
9	0	0	1	0	0	1	1	0			
10	0	0	1	0	1	0	1	1			8
11	0	0	1	0	1	1	1	1			9
12	0	0	1	1	0	0	0	0	svítí	dvojitý	0 až 7
13	0	0	1	1	0	1	0	0			
14	0	0	1	1	1	0	0	1			8
15	0	0	1	1	1	1	0	1			9
16	0	1	0	0	0	0	1	0	svítí	norm. ZN	0 až 7
17	0	1	0	0	0	1	1	0			
18	0	1	0	0	1	0	1	1			8
19	0	1	0	0	1	1	1	1			9
20	0	1	0	1	0	0	1	0	svítí	blik ZN	0 až 7
21	0	1	0	1	0	1	1	0			
22	0	1	0	1	1	0	1	1			8
23	0	1	0	1	1	1	1	1			9
24	0	1	1	0	0	0	1	0	svítí	blik. kurz.	0 až 7
25	0	1	1	0	0	1	1	0			
26	0	1	1	0	1	0	1	0			8
27	0	1	1	0	1	1	1	1			9
28	0	1	1	1	0	0	0	0	svítí	dvojitý	0 až 7
29	0	1	1	1	0	1	0	0			
30	0	1	1	1	1	0	0	1			8
31	0	1	1	1	1	1	0	1			9
	1	X	X	X	X	X	1	1			
	Horiz. zatmění HZ	$\overline{BL}$ 1,5 Hz	D_7	D_6	8L	1L	$\overline{D2}$	ZAT			
			data		čítač						

$tma = \overline{BL} \cdot \overline{D7} \cdot D6 + \overline{BL} \cdot D7 \cdot \overline{D6}$

blik. znak; blik. kurzor

```
JPR - 1 AND - 1 DISPLAY MEMORY TABLE
====================================
38 38 38 38 38 38 38 38 38 38 38 38 38 39 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27

38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61 62 63 64 65 66 67

38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F A0 A1 A2 A3 A4 A5 A6 A7

38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF E0 E1 E2 E3 E4 E5 E6 E7

39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27

39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61 62 63 64 65 66 67

39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F A0 A1 A2 A3 A4 A5 A6 A7

39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF E0 E1 E2 E3 E4 E5 E6 E7

3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27

3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61 62 63 64 65 66 67

3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F A0 A1 A2 A3 A4 A5 A6 A7

3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A 3A
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF E0 E1 E2 E3 E4 E5 E6 E7

3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27

3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61 62 63 64 65 66 67

3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F A0 A1 A2 A3 A4 A5 A6 A7

3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B 3B
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF E0 E1 E2 E3 E4 E5 E6 E7

3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27

3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61 62 63 64 65 66 67

3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F A0 A1 A2 A3 A4 A5 A6 A7

3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C 3C
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF E0 E1 E2 E3 E4 E5 E6 E7

3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27

3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61 62 63 64 65 66 67

3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F A0 A1 A2 A3 A4 A5 A6 A7

3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D 3D
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF E0 E1 E2 E3 E4 E5 E6 E7
```

zobrazen. Nakonec ještě jednou zopakuji, jak se kóduje typ znaku pro AND-1. Tak např. číslice 3 je v 6bitovém kódu vyjádřena takto: 11 0011 (33H). Přidáme-li dva nejvyšší bity a zapíšeme-li tento kód do paměti AND-1, bude zobrazení znaku:

0011 0011 (33H)	znak 3,
0111 0011 (73H)	znak 3, který celý bliká,
1011 0011 (B3H)	znak 3, pod nímž bliká kurzor,
1111 0011 (F3H)	znak 3 dvojnásobné šířky.

Chceme-li zobrazit pouze kurzor, zapíšeme do paměti znak SPACE (mezera) s příznakem blikajícího kurzoru: 1010 0000 (A0H).

Seznam součástek

Odpory (TR 191 ± 10 %, označení K)

R_1	560 Ω	R_8	68 Ω
R_2	3,9 kΩ	R_{15}, R_{16}, R_{21}, R_{22}, R_{34}, R_{35}	4,7 kΩ
R_3, R_7, R_9 až R_{14}	1 kΩ	R_{19}	47 kΩ
R_4, R_{31}	330 Ω	R_{20}	15 kΩ
R_5, R_{17}, R_{18}	680 Ω	R_{23} až R_{29}, R_{32}, R_{33}	10 kΩ
R_6	220 Ω	R_{30}	4,7 Ω

Kondenzátory

C_1 až C_{21}	22 nF, TK 783
C_{22}, C_{25}	4,7 µF/6,3 V, TE 121
C_{23}, C_{24}	50 µF/6 V, TE 981
C_{25}	100 pF, TGL 5155-A/100/5/63
C_{27}	150 pF, TGL 5155-A/150/5/63
C_{28}	10 nF, TK 783
C_{29}	47 nF, TK 783
C_{30}	6,8 nF, TK 783
C_{31}	5 µF/15 V, TE 984

Polovodičové prvky

T_1	KSY62 (KSY21)
T_2	KC508
T_3	KSY71
T_4	KSY81 (KSY82, TR15)
T_5	KSY62 (KSY21, KSY71)
D_1 až D_8	KA206
A_2, D_6	MH7400
B_1, D_4	MH7404
D_3	MH7410
B_2	MH7490A
B_7, C_5, C_6, C_7, D_2	MH7493A
A_7, D_1, D_5	MH7474
C_1	MH7496
A_3	MH3205
A_4, A_5	MH3216
D_7	UCY7406
D_8	UCY74123
C_2	UCY74153
A_6, B_5, B_6	UCY74157
A_8, C_8	MH74164
C_3	MH74188
A_1	MHB2501
B_3, B_4	2114/250 ns

Ostatní součástky

K1	konektor FRB TY 517 62 11
K3	konektor FRB TY 513 30 11
L_1	kostra mf cívky (6PF 260 09 – TESLA Orava), Ø kostry 5 mm, jádro M4 × 0,5 × 6 mm; 27 závitů drátu o Ø 0,2 mm CuU (LCUA)
Tr_1	hrníčkové jádro o Ø 14 × 8 × 8 mm (205 517 0 05 150 Pramet Šumperk), hmota H22, primár *1–2* 12 závitů, sekundár *3–4* 48 závitů drátu o Ø 0,2 mm CuU (LCUA), Tr_1 připevnit mosazným šroubkem M3
K2	konektor BNC (pouze pro TVP, není však nutný, potřebný výstup je i na konektoru K3). K2 připevnit na úhelník

objímky pro IO: 24 vývodů A1
18 vývodů B3, B4
16 vývodů C3

deska s plošnými spoji AND-1

zobrazovací jednotka: TVP se síťovým transformátorem (Satelit), AZJ 462 zobrazovací jednotka TESLA Orava

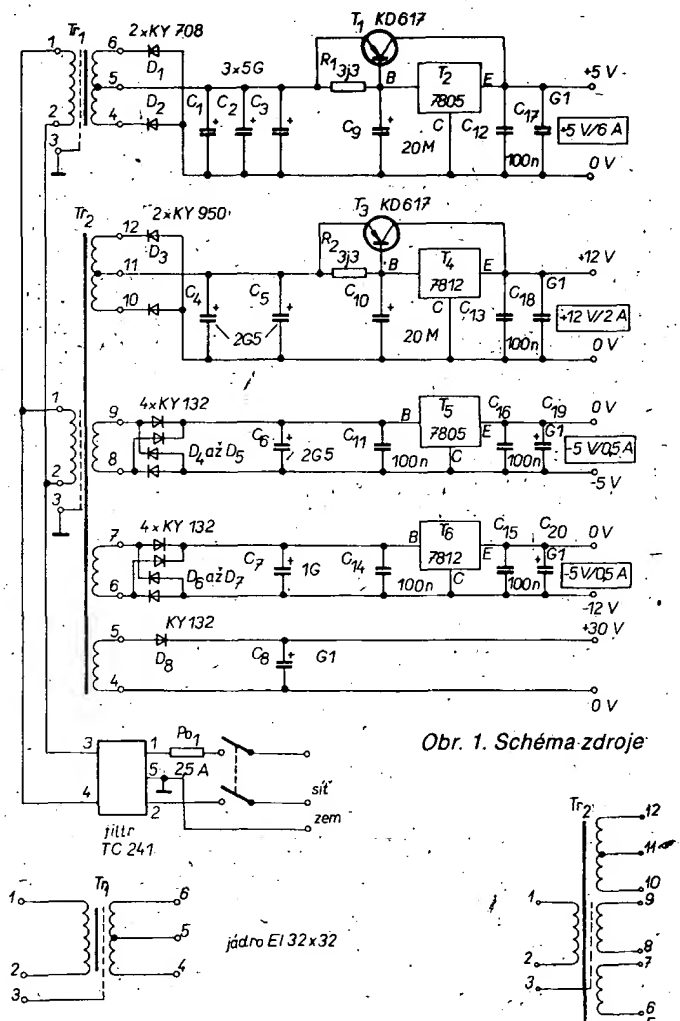
kabel AND-1 na AZJ 462, konektory TX 514 30 13 a WK 180 23, vodič VFDP 90 – 2 × 0,40 mm

JEDNOTKA ZDROJE A SBĚRNICE, JZS-1

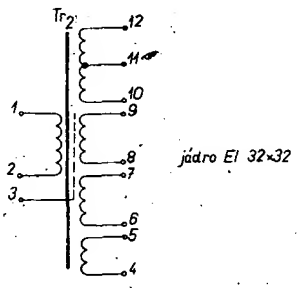
Jednotka JZS-1 zajišťuje mechanické vedení desek při zasouvání do konektorů sběrnice. Dále je v této jednotce napájecí zdroj a systémový panel, na němž je konektor pro napájení přídavných zařízení systémů. V létě 1982, kdy jsme připravovali toto číslo AR, byl vyroben pouze první vzorek této jednotky. Tento vzorek má jednoduchý napájecí zdroj, který je umístěn vzadu za deskami. Po prvních zkušenostech bude vývoj této jednotky dokončen. Napájecí zdroj bude samostatný a bude jej možno umístit buď vedle desek nebo za desky. Zapojení zdroje bude doplněno jištěním proti zkratu a přepětí na výstupu. Vodítka desek, držáky vodítek a další díly budou navrženy tak, aby nebyly tolik náročné na obrábění. Pro první „hraní" a aplikace JPR-1 však současná mechanika JZS-1 určitě vyhoví.

Napájecí zdroj

Schéma napájecího zdroje je na obr. 1. Zapojení jednotlivých regulátorů je co nejjednodušší. Na obr. 2 a 3 jsou navíjecí předpisy pro Tr_1 a Tr_2. Zdroj má i výstup pomocného napětí pro programátor pamětí EPROM. Programátor však musí mít svůj stabilizátor na napětí 26 V. Zdroj není postaven na desce s plošnými spoji a součástky patřící k jednotlivým regulátorům jsou připájeny přímo na vývody výkonových tranzistorů T_1 a T_3 a regulátorů MA78XX. Pouze filtrační kondenzátory C_1 až C_8 jsou na „čtverečkové" univerzální desce s plošnými spoji. Na stejné desce jsou i diody D_4 až D_8. Diody D_1 až D_3 jsou na chladiči. Výkonové prvky zdrojů +5 V a +12 V jsou na zadním chladiči jednotky JZS-1. Tranzistory T_1 a T_3 musí být připevněny k chladiči izolovaně, protože chladič je vlastně zemí zdroje i celé mechaniky a systému. Výkonové prvky zdrojů +5 a −5 V (T_5 a T_6) mají každý svůj chladič, aby je nebylo nutné izolovat od chladiče. Tyto dva menší chladiče jsou na desce kondenzátorů, kondenzátory jsou na spodní straně desky a chladiče a diody na horní. Chladiče nejsou na obrázcích vidět, protože jsou pod transformátory. Spoje mezi filtry a regulátory jsou zakončeny autokonektory, aby bylo možné sestavit regulátory na chladiči, a pak vše jednoduše propojit. Stejným způsobem je



Obr. 1. Schéma zdroje



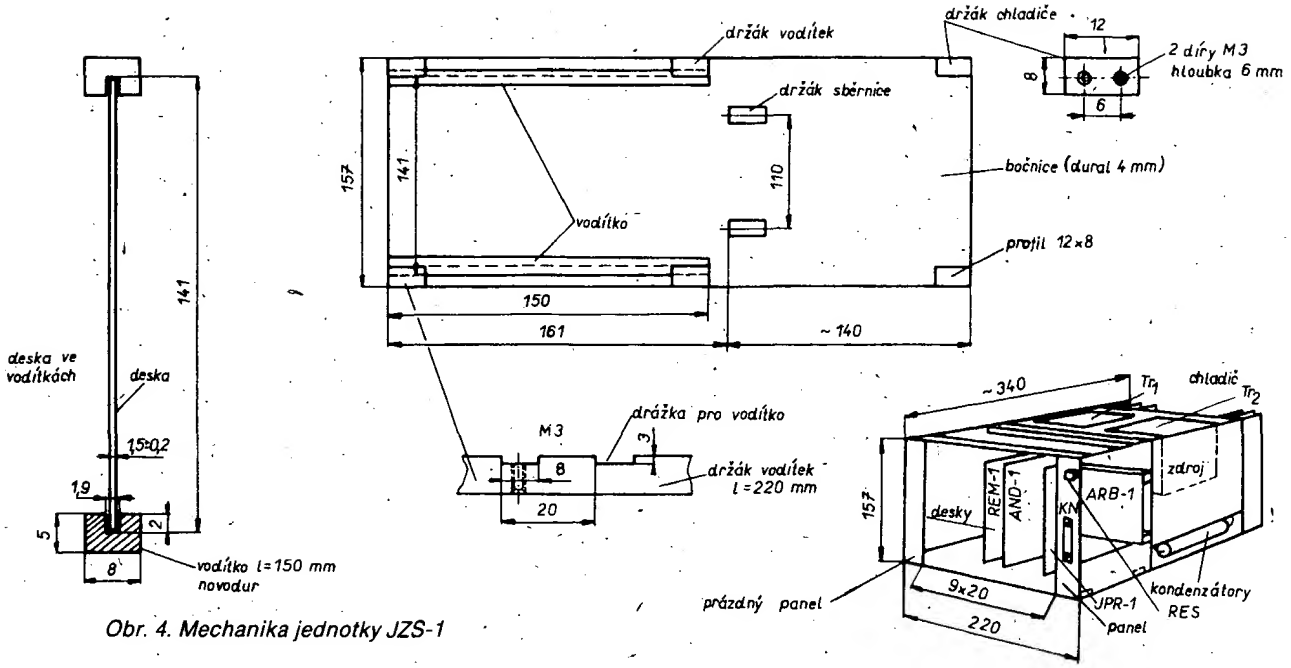
vinutí	napětí	proud	závitů	ϕ drátu
1–2	220 V	0,28 A	1020	0,355
3	stínění	fólie Cu ± 0,1		
4–5	9,6 V	6,5 A	45	1,6
5–6	9,6 V	6,5 A	45	1,6

vinutí 1–2 s proklady

▲ Obr. 2. Navíjecí předpis pro Tr_1

vinutí	napětí	proud	závitů	ϕ drátu
1–2	220 V	0,27 A	1020	0,355
3	stínění	fólie Cu ± 0,1		
4–5	35 V	0,1 A	160	0,2
6–7	16,5 V	0,7 A	77	0,6
8–9	10 V	0,7 A	47	0,6
10–11	15,5 V	2,2 A	71	1,06
11–12	15,5 V	2,2 A	71	1,06

Obr. 3. Navíjecí předpis pro Tr_2 ▶

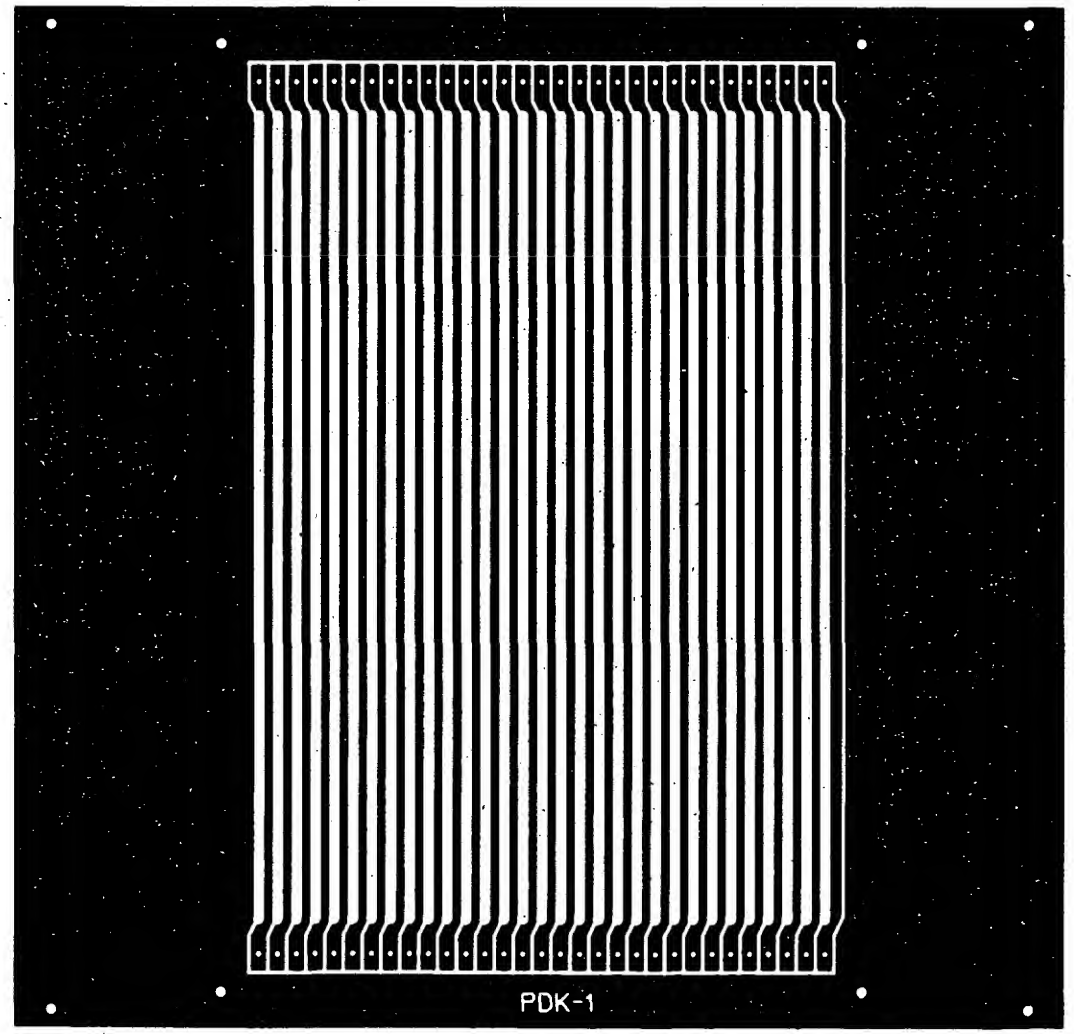


Obr. 4. Mechanika jednotky JZS-1

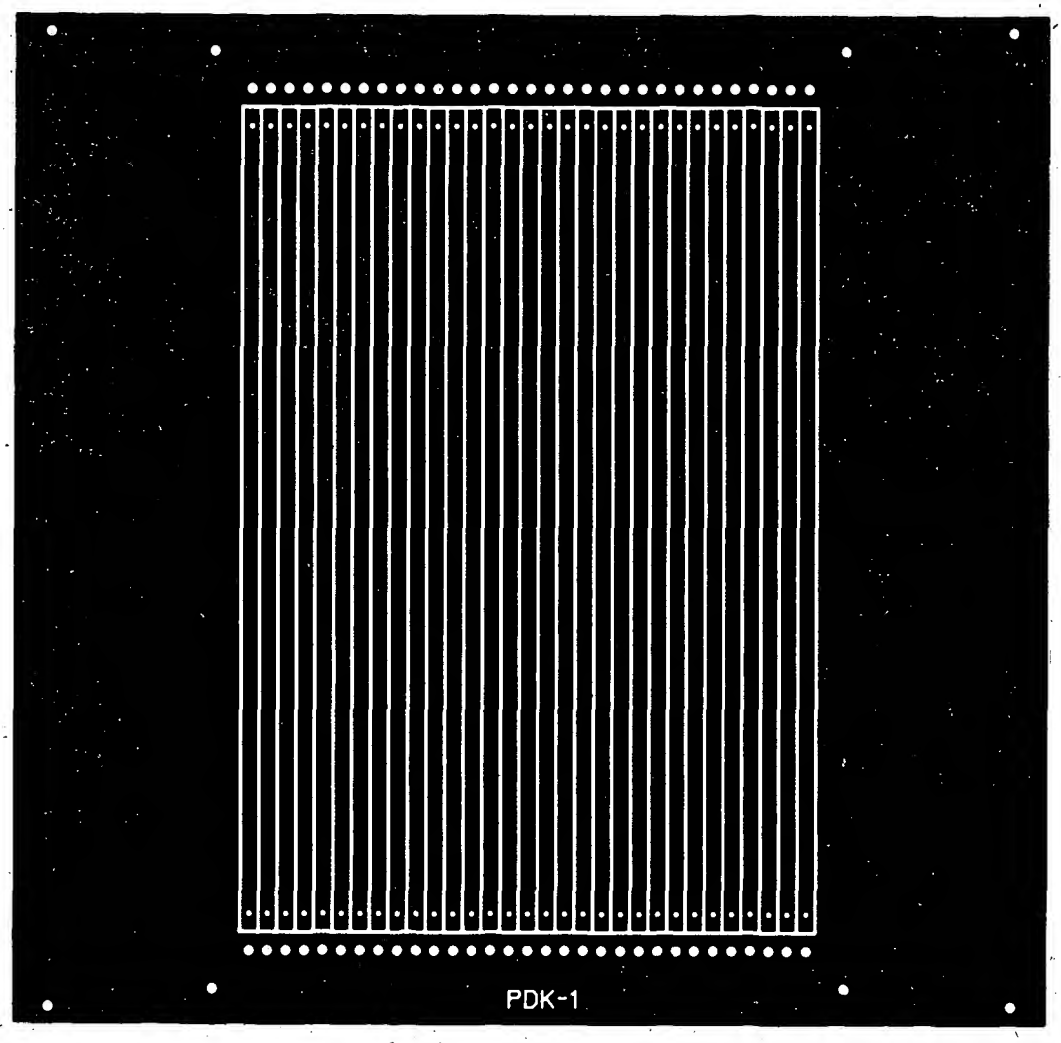
přivedeno výstupní napětí zdrojů na sběrnici ARB-1.

Síťová vanička, pojistka, filtr a malá svorkovnice (čokoláda) jsou připevněny na chladiči. Přes svorkovnici se vede síťové napětí na transformátory. Transformátory jsou připevněny na bočnice, každý na jednu.

Nepočítáme-li s tím, že budeme oživované desky napájet z popsaného zdroje, pak skutečně zdroj, i bez pojistek, vyhoví pro systém JPR-1. Zdroje ve vzorku byly zkoušeny na krátký zkrat a „vydržely" ho. Pozor však na to, že u mikroprocesorových systémů je jedno, vydrží-li nadměrnou zátěž zdroj, spíše je důležité, vydrží-li nepřítomnost jednoho napájecího napětí integrované obvody MOS nebo bipolární obvod 8228 (ten se obvykle zničí, chybí-li napětí +5 V). Chcete-li proto doplnit zdroj jištěním, musíte hlídat všechna napětí současně, tj. vypne-li jedna pojistka, musí samočinně vypnout všechny. Hlídat sekvence nárůstu a výpadku jednotlivých



Obr. 5. Deska s plošnými spoji PDK-1, horní strana

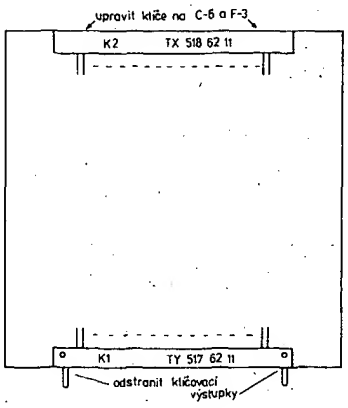


Obr. 6. Deska s plošnými spoji PDK-1, spodní strana

napájecích napětí není u JPR-1 nutné – není to ostatně zvykem ani u zahraničních malých systémů. U pamětí DRAM 4116 hlídám zvětšování napětí +12 V jednoduchým obvodem, který má jako referenci napětí −5 V. Tento obvod je však až na desce 32K DRAM, která je ve vývoji.

Mechanika jednotky

Mechanika jednotky je na obr. 4. Vodítka jsou vyfrézována z novoduru tloušťky 5 mm. Jednotka má dvě bočnice z duralu (4 mm). Bočnice jsou spojeny osmi profily (8 × 12 mm, *l* = 220 mm), které jednotce zajistí tuhost i při velké váze transformátorů. Čtyři profily se nazývají držák vodítek a jsou každých 20 mm profrézovány a provrtány pro připevnění vodítek. Vodítka se dají svrtat až po zasazení do drážek držáků.



Obr. 6a.

Další dva profily (držák sběrnice) slouží k připevnění desky ARB-1. Otvory v desce ARB-1 musí být větší, aby bylo možno nastavit konektory K1 a K8 přesně proti vodítkům (po sestavení jednotky). Nasuneme desky do pozic 1 a 8 sběrnice a pak teprve utáhneme šroubky, které drží sběrnice na držácích. Místo podložek je lepší pod šroubky použít pásek délky 200 mm (kuprextit), který má tři otvory a „přitáhne" celou sběrnici po celé délce.

V zadní části jednotky je napájecí zdroj. Vpředu jsou na každé straně dva malé panely. Levý jen vyplňuje prázdné místo a pravý je tak zvaný systémový. Může na něm být tlačítko RESET (není-li na klávesnici ANK-1), dále konektor (nebo raději několik menších konektorů), který slouží pro napájení přídavných zařízení systému JPR-1. Sem se připojuje i napájecí kabel pro AZJ 462 nebo pro klávesnici ANK-1 a v budoucnu pro další periférie.

Prodlužovací deska PDK-1

Prodlužovací deska je dobrým pomocníkem při hledání závad. I když nerad měřím pomocí „prodlužovačky" (prodloužení přívodů zanáší někdy do systému další chyby, hlavně u větších desek), přece jen je to někdy nutné. Deska je sice také oboustranná, ale nepotřebuje prokovené díry. Deska PDK-1 je kratší (135 mm), aby i deska, na které se měří, byla alespoň trochu vedena vodítkem. Na jedné hraně desky je zásuvka, na druhé zástrčka (obr. 5, 6).

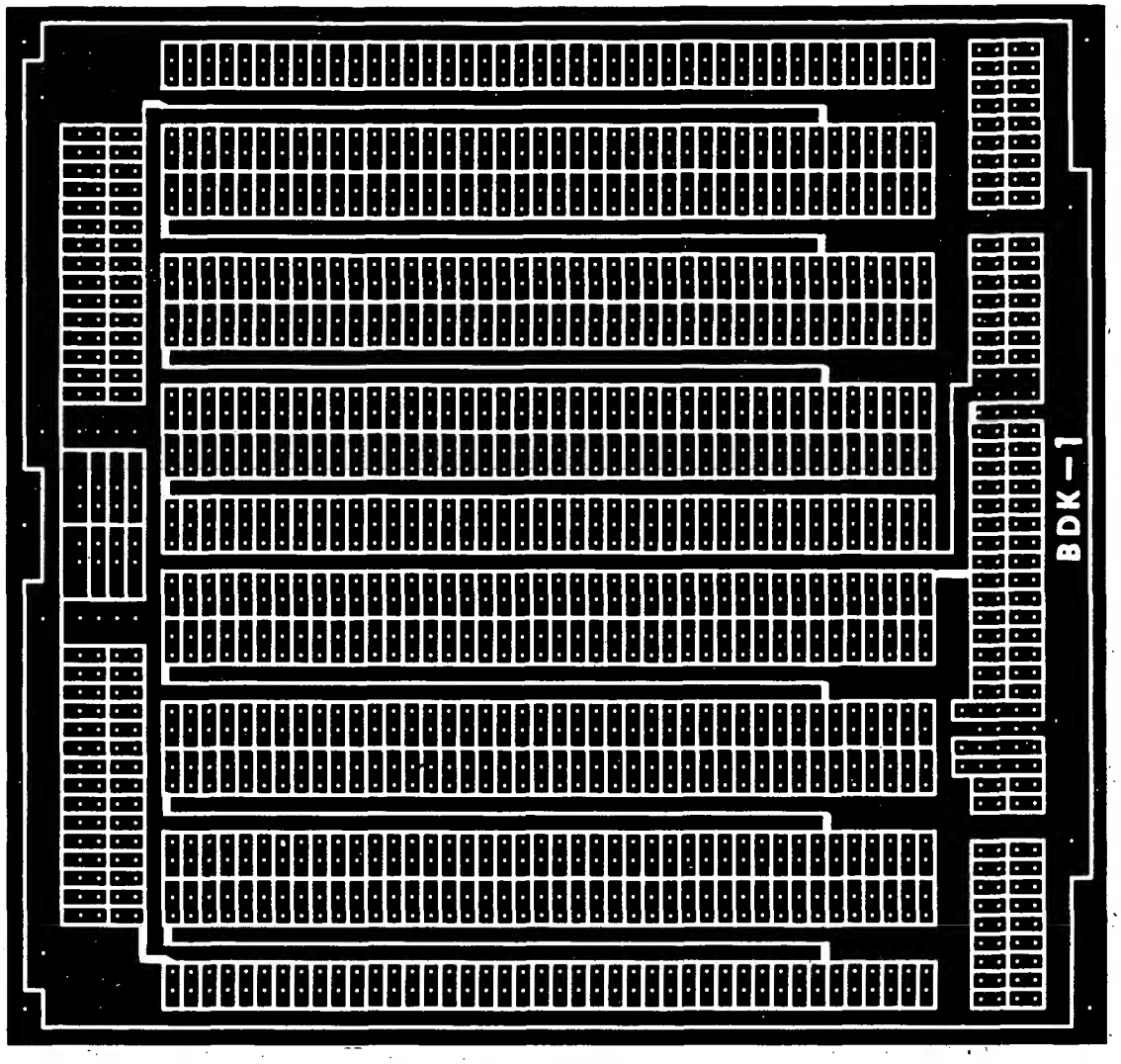
Univerzální deska BDK-1

Tak konečně i jedna jednostranná deska s plošnými spoji v systému JPR-1. Deska BDK-1 je univerzální a slouží k ověřování zapojení dalších desek systému. Na desku lze umístit i obvody s širokou roztečí vývodů (24, 28 a 40vývodové obvody), a to buď doprostřed desky nebo i jinam, oželíme-li místo, kam by se vešel malý obvod. Deska BDK-1 má na konektoru K1 (obr. 7, na němž je rozmístění standardních konektorů JPR-1) již zem a +5 V. Po stranách konektoru je místo na blokovací kondenzátory. Mezi K2 a K3 je místo na odpory, připojené jedním koncem na +5 V.

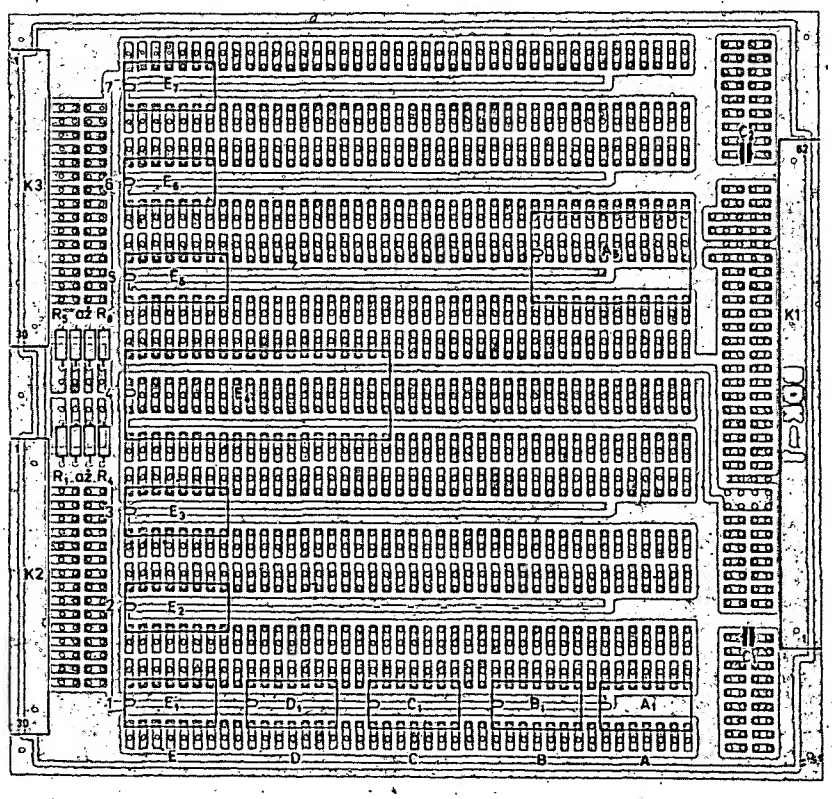
Při ověřování zapojuji desky drátem LCUA se samopájitelnou izolací.

Věřím, že tato deska umožní i vám postavit si něco k systému JPR-1. A také věřím, že se pak na stránkách AR objeví i váš článek, který bude zajímat všechny uživatele systému JPR-1.

Když jsem se svými kolegy vyvinul počítač JPR-12 a zavedl jeho výrobu, měli jsme na konferencích o systému SAPI heslo: k Jé Pé eR jde připojit všechno. Tak ať nyní platí toto heslo i o JPR-1.



Obr. 8. Deska s plošnými spoji BDK-1



Obr. 7. Konektory na desce BDK-1

Seznam součástek

Napájecí zdroj

Diody

D_1	KY708
D_2	KY708
D_3	KY950
D_4 až D_8	KY132/60

Tranzistory

T_1, T_3	KD617
T_2, T_5	MA7805
T_4, T_6	MA7812

Odpory

R_1, R_2	3,3 Ω, TR 223

Kondenzátory

C_1, C_2, C_3	5000 µF, TE 674
C_4, C_5	2500 µF, TE 676
C_6	2500 µF, TE 674
C_7	1000 µF, TE 675
C_8	100 µF, TE 986
C_9, C_10	20 µF, TE 986
C_{11} až C_{16}	100 nF, TK 783
C_{17} až C_{20}	100 µF, TE 984

Ostatní součástky

síťový spínač, síťová vanička, držák pojistky, pojistka 2,5 A, filtr TC 241, slídové podložky pod tranzistory T_1 a T_3, šroubky, vodiče, pájecí očka, automobilové konektory, chladič velký (T_1, T_2, T_3 a T_4) a chladič malý – 2 ks (T_5 a T_6)

Tr_1 a Tr_2 transformátory, jádro EI32 × 32

Mechanika jednotky
deska sběrnice ARB-1 sestavená, 2 ks bočnice, 4 ks držák vodítek, 16 ks vodítko, 2 ks držák sběrnice, 2 ks držák chladiče
Deska PDK-1
K1 TY 517 62 11
K2 TX 518 62 11
Deska s plošnými spoji PDK-1
Deska BDK-1
K1 TY 517 62 11
K2 a K3 TY 513 30 11
deska s plošnými spoji BDK-1

Literatura

[2] *Titus, J. A. a kol.:* Seriál článků (rubrika Micro Data Stack). Computer Design 1977, 1978, 1979 a 1980.

[3] *Kane, J.; Osborne, A.:* An introduction to microcomputers. Adam Osborne & Associates, Inc., Berkeley: USA 1978.

[4] *Švecová, M.:* Kódově nezávislý psací stroj řízený mikroprocesorem. Automatizace č. 2/1981.

[5] *Titus, J. A.:* TRS-80 interfacing. Howard W. Sams & Co., Inc.: USA 1980.

[6] *Martin, D. P.:* Microcomputer design. Martin research 1976.

[7] *Pasahow, E. J.:* Microcomputer Interfacing for electronics technicians. Greg Division, McGraw-Hill Book Company: New York 1981.

[8] *Černoch, M. a kol.:* Technické prostředky a funkce mikroprocesoru 8080A. Sdělovací technika č. 12/1981.

[9] Přílohy časopisu Automatizace: Mikroprocesorové systémy v řídící technice (1980); *Smejkal, L.:* Kurs programování s mikroprocesorem 8080 (1981 a 1982).

[10] Perspektivní součástková základna pro elektroniku. Materiál vydaný FMEP a TESLA Rožnov.

[11] Technické zprávy TESLA Rožnov.

[12] Firemní literatura TI, INTEL, Motorola, AMD, DEC, Sycor, Pertec, Hughes, Søren T. Lyngsø, Sumagrafic, Schlummberger a dalších.

[13] Časopisy z let 1972 až 1982: Elektronik (NSR), Electronics, Computer Design, Electronics Design, Microcomputing, Byte (USA). Sdělovací technika a Automatizace (ČSSR).

[14] Katalogy pasívních a konstrukčních součástek. TESLA Lanškroun.

[15] *Carlstrom, R.:* Designing with 8080 microprocessor. Popular Electronics, seriál článků 1981 a 1982.

IV. Programování mikropočítače JPR-1

Ing. Tomáš Smutný

Počítač bez programu je jako auto bez benzinu a v případě JPR-1 je to ještě horší. Jednoduchá konstrukce přídavných zařízení (např. klávesnice a displeje) vyžaduje totiž určité programové vybavení – program v paměti ROM tak vlastně nahrazuje řadu součástek. Této části programového vybavení se říká firmware (podobně jako hardware a software) a teprve po jeho napsání a odladění je počítač JPR-1 schopen dvou základních funkcí, tj. přijmout znak z klávesnice a zobrazit znak na displeji.

Po návrhu a odladění firmware lze přistoupit k návrhu základního programu mikropočítače, který usnadní přípravu a ladění programů přímo ve strojovém kódu mikroprocesoru 8080. Tímto programem je monitor. A nechceme-li zůstat na úrovni typických jednodeskových mikropočítačových stavebnic, např. SDK-85, zatoužíme zpravidla po interpreteru jazyka BASIC.

To vše je potřeba doplnit o možnost připojit tiskárnu a cokoli, co umožní uschovat odladěné programy (může to být např. snímač a děrovač děrné pásky nebo kazetový magnetofon).

Dá-li vám k tomu konstruktér fungující (podle jeho představ) počítač a prostor paměti 4K EPROM, jste přesně v situaci, do níž mne bratr dostal před necelým rokem.

Problém, zda bylo první vejce nebo slepice, se mi již nezdál tak vtipný jako dříve. Zejména proto, že ještě nebyly k dispozici ani popisovaný přípravek TST-03 ani simulátor EPROM.

Vznikl tedy nový problém: Jak napsat program pro počítač, který neumí ani přečíst znak z klávesnice, ani jej zobrazit a nemá pochopitelně žádného předchůdce k případnému opisování.

Jak to dopadlo se dozvíte na dalších stránkách. Zbývá jen omluvit se těm, kdož o programování mikropočítačů vědí málo nebo nic. Byli jsme totiž nuceni volit mezi dvěma variantami: Buď začít od základů programování mikroprocesoru 8080, nebo skočit přímo tam, kde končí popis konstrukce a začíná programové vybavení. Zvolili jsme druhou variantu a ty zklamané mohu ujistit, že se k teorii, bude-li zájem, ještě v budoucnu vrátíme. K autu dostanete také nejprve návod k obsluze a teprve čas vás přinutí vrátit se k teorii.

Základní programové vybavení JPR-1

Místo základní by bylo možno napsat minimální – představuje totiž realizaci požadavků, od nichž jsme nechtěli ustoupit při omezení paměti na 4K EPROM, realizované s obvody 2708 na desce procesoru JPR-1. To dovoluje, s omezením použitelné paměti RAM asi na 700 bytů, pracovat v jazyce BASIC již při dvojdeskové variantě počítače (desky JPR-1 a AND-1).

I když by se zdálo, že je to programové vybavení „chudého amatéra", uvidíte sami, že počítač je velmi výkonný a postačí dokonce pro řadu aplikací i při profesionálním použití. Naučíte-li se s ním pracovat a nebudete-li se bát poněkud většího množství práce, než kolik je jí třeba u větších počítačů, osvědčí se vám Mikro BASIC a Mikro monitor jako výborná průprava pro váš další programátorský růst. Programové vybavení řady Mikro, jak jsme ho u JPR-1 nazvali, odstraňuje totiž pracnost práce ve strojovém kódu s hexadecimální klávesnicí, umožňuje pracovat s vyšším programovacím jazykem, a zároveň dává programátorovi možnost, zachovat si přehled o tom, jak jsou data a např. textové řetězce uloženy v paměti, jak pracuje počítač s proměnnými, co je to indexovaná proměnná atd.

V provedení Mikro je také assembler 8080, zpětný assembler, textový editor atd. Všechny programy této řady mají délku 2 až 4K a samozřejmě příslušně omezené funkce a parametry.

Kromě úrovně Mikro má však JPR-1 zhruba další dvě úrovně složitosti programového vybavení. Střední úroveň zahrnuje programy s délkou 4K až 6K a bude mít také střední, standardní komfort. Velmi výkonné programy o délce 8K až 16K budou obsaženy v třetí úrovni a budou určeny zejména pro profesionální použití počítače JPR-1.

Výpis programu

Pro ty, kdož chtějí pozorně sledovat následující popis, bude užitečné seznámit se s formátem výpisu programu.

Celý program byl napsán v assembleru 8080 pomocí textového editoru a přeložen do strojového kódu pomocí assemblerovského překladače.

V textu není celý program otištěn, a proto je velmi důležitý první sloupec vlevo, který udává pořadové číslo řádku ve zdrojovém textu. Druhý sloupec udává absolutní adresu v paměti počítače, pochopitelně v hexadecimálním tvaru.

Třetí, nejdůležitější sloupec obsahuje výsledný, hexadecimální kód programu generovaný překladačem při překladu zdrojového textu. Je to tedy zároveň obsah pamětí EPROM na desce procesoru JPR-1. Celý tento výpis je otištěn v hexadecimálním tvaru.

Dále následuje zdrojový text v symbolickém jazyce, assembleru, se sloupcem návěští, mnemonickým názvem kódu instrukce a sloupcem parametrů. Poznámky nejsou použity jednak pro úsporu místa a jednak nezaškodí, „prokoušete-li" se v případě hlubšího zájmu programem sami.

Hlavní části programu

Základní programové vybavení bude popisováno podle jednotlivých funkčních celků, jimiž jsou: programová obsluha klávesnice a displeje, interpreter BASIC a monitor. Připojení ostatních periferií předpokládá popis hardware a proto necháme tento problém raději na jindy.

Jednotlivé části programu jsou použitelné univerzálně. Část týkající se klávesnice a displeje lze použít ve spojitosti s libovolným programem pro JPR-1 a naopak interpreter BASIC a monitor je možno použít v jakémkoliv mikropočítači s CPU 8080 nebo Z80; pak je však třeba pozměnit adresy podprogramů pro vstup a výstup znaku.

Programová obsluha klávesnice

Alfanumerická klávesnice ANK-1 je zcela pasívním prvkem. Zatímco běžná klávesnice obsahuje čítač, dekodér a matici tlačítek, má ANK-1 pouze matici tlačítek. Postupné „ohledávání" matice tlačítek je u běžné klávesnice zabezpečeno vlastní elektronikou, u ANK-1 i tuto funkci musí převzít program. Dále je třeba zjistit, bylo-li stisknuto současně tlačítko „shift", nebyla-li stisknuta současně jiná dvě tlačítka a zabezpečit vstup pouze jednoho znaku při každém novém stisku tlačítka s vyloučením vlivu zákmitů při stisku.

Mimo to je nutné postarat se o rozsvícení indikace tlačítka „shift" a indikovat

každý správný stisk tlačítka zvukovým signálem (to proto, že zdvih tlačítka je velmi malý).

Aby vše bylo ještě složitější, musí program zastoupit ještě funkci dekodéru pořadového čísla tlačítka na kód ASCII (to proto, že bylo zvoleno standardní rozložení znaků na klávesnici, QWERTY, a tlačítka jsou propojena maticově).

Pro ty, kdož se nechtějí vracet k popisu konstrukce a zapojení klávesnice ANK -1 zopakuji, že matice tlačítek má organizaci 5 × 8 s tím, že 5 řádků je připojeno na výstupní port 0 a 8 sloupců na vstupní port 0. Oba porty, adresované jako paměť, mají adresy v rozsahu 2400H až 27FFH (hexadecimálně).

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0


19690*                  ;MIKRO BASIC JPR-1
19700                   ;**********************************
19710                   ;
19720                   ;    KLAVESNICE    JPR-1
19730                   ;
19740                   ;**********************************
19750                   ;
19760*                  ;ZACATEK
19770                   ;
19780 0CF8 C5       CI:     PUSH  B
19790 0CF9 D5               PUSH  D
19800 0CFA E5               PUSH  H
19810 0CFB 0E80     KLIN10: MVI   C,80H
19820 0CFD 1605     KLIN20: MVI   D,5
19830 0CFF 061E             MVI   B,1EH
19840 0D01 2600             MVI   H,0
19850*                  ;
19860*                  ;GENERACE SIGNALU PRO SLOUPCE
19870*                  ;
19880 0D03 3A0020   KLIN30: LDA   PORT24
19890 0D06 E6E0             ANI   0E0H
19900 0D08 B0               ORA   B
19910 0D09 320024           STA   2400H
19920 0D0C 78               MOV   A,B
19930 0D0D 37               STC
19940 0D0E 17               RAL
19950 0D0F E61F             ANI   1FH
19960 0D11 47               MOV   B,A
19970 0D12 3A0024           LDA   2400H
19980 0D15 FEFF             CPI   0FFH
19990 0D17 C44E0D           CNZ   KLIN50
20000 0D1A 15               DCR   D
20010 0D1B C2030D           JNZ   KLIN30
20020                   ;
20030                   ;PROBEHL CELY CYKLUS , 5 SLOUPCU
20040                   ;
20050 0D1E CD740D           CALL  SHOF
20060 0D21 79               MOV   A,C
20070 0D22 FE01             CPI   1
20080 0D24 CA320D           JZ    KLIN40
20090 0D27 DAFD0C           JC    KLIN20
20100 0D2A FE80             CPI   80H
20110 0D2C CA430D           JZ    KLIN90
20120 0D2F C3FB0C           JMP   KLIN10
20130                   ;
20140*                  ;PLATNY ZNAK A NAVRAT
20150                   ;
20160 0D32 7D       KLIN40: MOV   A,L
20170 0D33 84               ADD   H
20180 0D34 21890D           LXI   H,TABZN
20190 0D37 4F               MOV   C,A
20200 0D38 0600             MVI   B,0
20210 0D3A 09               DAD   B
20220 0D3B CDF50D           CALL  PIPO
20230 0D3E 7E               MOV   A,M
20240 0D3F E1               POP   H
20250 0D40 D1               POP   D
20260 0D41 C1               POP   B
20270 0D42 C9               RET


ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0


20280*                  ;
20290*                  ;KLAVESNICE V KLIDU
20300*                  ;
20310 0D43 0E90     KLIN90: MVI   C,90H
20320 0D45 0D       KLIN99: DCR   C
20330 0D46 C2450D           JNZ   KLIN99
20340 0D49 0E00             MVI   C,0
20350 0D4B C3FD0C           JMP   KLIN20
20360                   ;
20370                   ;PROHLEDANI RADKU
20380                   ;
20390 0D4E 1E08     KLIN50: MVI   E,8
20400 0D50 0F       KLIN60: RRC
20410 0D51 D4590D           CNC   KLIN70
20420 0D54 1D               DCR   E
20430 0D55 C2500D           JNZ   KLIN60
20440 0D58 C9               RET
20450                   ;
20460                   ;SESTAVENI KODU KLAVESY
20470                   ;
20480 0D59 F5       KLIN70: PUSH  PSW
20490 0D5A 7A               MOV   A,D
20500 0D5B 3D               DCR   A
20510 0D5C 07               RLC
20520 0D5D 07               RLC
20530 0D5E 07               RLC
20540 0D5F E638             ANI   38H
20550 0D61 1D               DCR   E
20560 0D62 B3               ORA   E
20570 0D63 1C               INR   E
20580 0D64 FE07             CPI   7
20590 0D66 CA6D0D           JZ    KLIN80
20600 0D69 6F               MOV   L,A
20610 0D6A 0C               INR   C
20620 0D6B F1               POP   PSW
20630 0D6C C9               RET
20640*                  ;
20650*                  ;PRIZNAK SHIFT
20660*                  ;
20670 0D6D 2628     KLIN80: MVI   H,28H
20680 0D6F CD800D           CALL  SHON
20690 0D72 F1               POP   PSW
20700 0D73 C9               RET
20710*                  ;
20720*                  ;SIGNALISACE SHIFT
20730*                  ;
20740 0D74 3A0020   SHOF:   LDA   PORT24
20750 0D77 E6D0             ANI   0D0H
20760 0D79 320020   SHOF10: STA   PORT24
20770 0D7C 320024           STA   2400H
20780 0D7F C9               RET
20790 0D80 3A0020   SHON:   LDA   PORT24
20800 0D83 F620             ORI   20H
20810 0D85 C3790D           JMP   SHOF10
20820 0D88 C9               RET
20830                   ;
20840                   ;TABULKA ASCII
20850                   ;
20860 0D89 30500D08 TABZN:  DB    '0P',0DH,08H,'1QA'
           315141


ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0


20870 0D90 0E394F4C         DB    0EH,'9OL 2W'
           203257
20880 0D97 535A3849         DB    'SZ8IKM3EDX'
           4B4D3345
           4458
20890 0DA1 37554A4E         DB    '7UJN4RFC6Y'
           34524643
           3659
20900 0DAB 48423554         DB    'HB5TGV>'
           475629
20910*                  ;
20920*                  ;KODY PRI SHIFT
20930*                  ;
20940 0DB2 7F0D0821         DB    07FH,0DH,08H,'!0A'
           5141
20950 0DB8 0E28233D         DB    0EH,'(#= ?WSZ'
           203F5753
           5A
20960 0DC1 3E402A2E         DB    '>@*. "EDX<:'
           22454458
           3C3A
20970 0DCB 2D2C2452         DB    '-,$RFC',27H,'/+;^TGV'
           4643272F
           2B3B5E54
           4756
20980                   ;
20990                   ;PIPNUTI
21000                   ;
21010 0DD9 C5       PIP:    PUSH  B
21020 0DDA 3A0020           LDA   PORT24
21030 0DDD F640             ORI   40H
21040 0DDF 320024           STA   2400H
21050 0DE2 0E42             MVI   C,42H
21060 0DE4 0D       PIP20:  DCR   C
21070 0DE5 C2E40D           JNZ   PIP20
21080 0DE8 E6B0             ANI   0B0H
21090 0DEA 320024           STA   2400H
21100 0DED 0E50             MVI   C,50H
21110 0DEF 0D       PIP30:  DCR   C
21120 0DF0 C2EF0D           JNZ   PIP30
21130 0DF3 C1               POP   B
21140 0DF4 C9               RET
21150                   ;
21160                   ;ZAPIPANI
21170                   ;
21180 0DF5 114000   PIPO:   LXI   D,40H
21190 0DF8 CDD90D   PIPO10: CALL  PIP
21200 0DFB 1B               DCX   D
21210 0DFC 7A               MOV   A,D
21220 0DFD B3               ORA   E
21230 0DFE C2F80D           JNZ   PIPO10
21240 0E01 C9               RET
21250 0E02                  EJECT
```

Základní princip programové obsluhy klávesnice spočívá v tom, že počítač vysílá přes výstupní port signál log. 0 vždy pouze do jediného řádku, a zároveň (pomocí vstupního portu) zjišťuje, vyskytne-li se signál log. 0 v některém sloupci. Je-li tomu tak, je zřejmé, že v průsečíku řádku a sloupce bylo stisknuto tlačítko. Podívejme se však přímo na programové řešení, které je otištěno v assembleru.

Podprogram pro vstup znaku

Vstup do podprogramu je označen návěstím CI, podprogram začíná běžnou úschovou registrů do zásobníkové paměti (stack). Registr A, akumulátor, do stacku neukládáme. Na výstupu z podprogramu totiž právě v registru A očekáváme znak a starý obsah nás asi nebude zajímat. Uschovávat ostatní registry je nutné vzhledem k tomu, že všechny registry CPU budeme potřebovat a bylo by neefektivní uschovávat registry všude tam, kde budeme podprogram CI používat.

Pokračujeme uložením hexadecimálního údaje 80H do registru C. Vlastní velikost tohoto údaje je nezajímavá, slouží pouze jako příznak. Tento příznak je součástí mechanismu, který zajišťuje, že podprogram ignoruje trvale stisknuté tlačítko a při každém vyvolání si počká na uvolnění všech tlačítek. Jinými slovy: programem je realizován derivační obvod, reagující pouze na sestupnou hranu impulsu při stisku tlačítka. Celý mechanismus využívá obsahu registru C, ke kterému je vždy při testování celého pole klávesnice přičtena jednička, je-li určeno stisknuté tlačítko. Jedinou výjimkou je tlačítko „shift", při jehož stisku se jednička k C nepřičítá. Nutnou podmínkou pro úspěšné přijmutí a vyhodnocení znaku je stisk jediného tlačítka a obsah registru C se v tomto případě musí rovnat jedné.

Počáteční nastavení obsahu C na 80H způsobí, že při určení stisknutého tlačítka již při prvním ohledání pole bude obsah C=81H (nebo více). Podprogram pak bude tvrdošíjně čekat, až poprvé nalezne zcela uvolněná tlačítka, počká si okamžik (pomocí časové smyčky KLIN90) a nastaví příznak v C na nulu. Od tohoto okamžiku může být obsah C trvale nulový (není-li stisknuto žádné tlačítko), může být roven jedné (jde-li o platný znak) nebo dvěma (při současném stisku dvou kláves). Samozřejmě, že se může rovnat i třem, čtyřem atd., i když současně stisknout větší počet tlačítek než tři dokáže při běžném psaní jen kaskadér. A tak je do C znovu a znovu vkládána nula a čeká se na vytouženou jedničku. Pokračujme však dále.

Protože má výstupní port 0 připojeno pouze 5 řádků, připravíme si do registru D hodnotu 5 a určíme tomuto registru funkci čítače řádek klávesnice. Do registru B si pak připravíme první údaj, který vyšleme do řádkových vodičů. Hexadecimálně je tento údaj 1EH (pro méně zkušené – jde o binární číslo 0001 1110). Ještě si do registru H vložíme nulu jako příznak nestisknutého tlačítka „shift" a můžeme začít.

Hned u návěští KLIN30 nás však čeká nepříjemnost. Z portu 0 jsme si půjčili jen 5 bitů a nemůžeme jen tak jednoduše nechat rotovat nuly a jedničky přes zbývající 3 bity. Tam je připojena zvuková a světelná signalizace a klávesnice by blikala, pískala, prostě zlobila. Proto si pomůžeme pomocnou paměťovou buňkou PORT24, z níž si budeme hodnotu nejvyšších tří bitů portu 0 půjčovat.

Celý zázrak klávesnice pak spočívá v pěti cyklech, v nichž instrukcí STA 2400H posíláme na výstupní port 0 „nulu" do patřičného řádku, do registru B si připravíme novou hodnotu s nulou posunutou vlevo a pomocí instrukce LDA 2400H si ze vstupního portu 0 přečteme stav sloupců klávesnice.

Nalezneme-li v některém sloupci alespoň jeden nulový bit, odskočíme si na návěští KLIN50 a čítačem sloupců v registru E zjistíme, který to byl. Pak si z údajů v registrech D a E sestavíme pořadové číslo klávesy a to uložíme do registru L. Bylo-li pořadové číslo 7, což je klávesa „shift", změníme příznak v registru H na 28H a v každém případě pokračujeme v prohledávání celé klávesnice.

Po skončení pátého, posledního cyklu zjistíme již známým způsobem, jde-li o platný znak a převedeme pořadové číslo klávesy v registru L na kód klávesy v ASCII. Na rozloučenou ještě zapípáme, obnovíme registry a opustíme podprogram CI instrukcí RET.

Pro převod pořadového čísla klávesy na kód ASCII slouží tabulka TABZN. Pořadové číslo jednoduše přičteme k počáteční adrese tabulky, přečteme správný kód a ten vložíme do registru A. V případě, že byla stisknuta navíc klávesa „shift", poslouží hodnota 28H v registru H jako posuv adresy o polovinu tabulky, kde najdeme příslušné kódy.

Rozsvícení signalizace „shift", její zhasnutí a zapípání (akustická indikace) svěřím již k prozkoumání čtenářům.

Programová obsluha displeje

Alfanumerický displej AND-1 umí pouze transformovat obsah určité části paměti od adresy 3800H do množiny 64 znaků čtyř typů písma. Zapneme-li mikropočítač JPR-1 bez přítomnosti inteligentního programu, získáme názornou ukázku.

Úkolem programové obsluhy displeje je zbavit programátora nutnosti zabývat se adresami v paměti RAM displeje, umožnit mu psát znaky postupně vedle sebe a po řádcích, používat různé typy písma, nulovat programově displej a imitovat funkci tzv. rolování displeje. Vedle souboru 64 znaků ASCII pak bude displej rozumět znakům CR, LF a BS ze souboru tzv. řídicích znaků ASCII:

CR, carriage return neboli návrat vozíku znamená, že se kurzor vrátí na začátek rozepsaného řádku beze změny obsahu tohoto řádku. Další znaky tedy přepisují znaky původní.

LF, line feed neboli nový řádek způsobí přechod kurzoru na nový řádek v téže pozici na řádku, v níž byl naposled před LF. Potřebujeme-li bezpečně zařídit, aby text začínal na začátku nového řádku, je třeba použít kombinaci znaků CR a LF.

U mechanických dálnopisů trval návrat vozíků určitý čas, delší než byla doba posuvu válce na nový řádek. Proto se posílal nejprve znak CR, případně raději dva a potom LF. U alfanumerického displeje na pořadí znaků nezáleží.

Znak BS, back space, znamená návrat kurzoru o jednu pozici vlevo, bez změny předcházejícího znaku. Protože se u JPR-1 zatím neukázala nutnost, aby kurzor přecházel při BS také z první pozice na řádku na konec předcházejícího řádku, končí tento pohyb u AND-1 na témže řádku.

Rolování displeje znamená, že se po dopsání posledního znaku všechny řádky posunou o jeden nahoru. Obsah prvního, horního, jednoduše zmizí. Poslední řádka se uvolní, vynuluje. K pojmu vynuluje platí u displeje malá poznámka. Vynulování displeje, celé obrazovky nebo řádku, znamená v podstatě zaplnění mezerami. V paměti RAM displeje je v tomto případě hexadecimální kód 20H a kód 00H odpovídá u AND-1 znaku @

Program pro výstup znaku na displeji

Vstup do podprogramu je označen návěštím OUTDIS a podprogram opět začíná úschovou registrů do zásobníkové paměti (stack). V tomto případě jsme nezapomněli ani na registr A vzhledem k tomu, že znak může být testován až po jeho vypsání.

Potom je znak uložen ještě do registru C. V případě, že by bylo třeba změnit podprogram OUTDIS tak, aby vypisoval znak z registru C a ne z akumulátoru, je třeba instrukci změnit na MOV A, C. Výstup z registru C používají například programy mikropočítačů INTEL.

Následující volání podprogramu SESTAV předpokládá, že ve dvou buňkách paměti RAM s názvy POZICE a RADEK jsou uloženy údaje právě přístupného místa na obrazovce displeje.

V případě čísla pozice na řádku to může být údaj 0 až 39, číslo řádku pak může být 0 až 23. O správné nastavení těchto hodnot, jejich zvyšování a změny se stará celý programový mechanismus ovládání displeje.

Podprogram SESTAV pak jednoduše konvertuje tyto hodnoty na konkrétní adresu paměti RAM na desce AND-1. Je-li např. v obou buňkách počáteční hodnota 0, vrátí podprogram v registrech H, L adresu 3800H. Při maximálních hodnotách 39 a 23 bude v H, L hodnota 3DE7H, odpovídající poslednímu místu obrazovky v pravém dolním rohu. Celá konverze spočívá v přenesení údaje o čísle řádku do bitů 6 až 10, přenesení údaje o pozici v řádku do bitů 0 až 5 a nastavení bitů 11 až 15 na hodnotu 00111.

Další úsek programu až po návěští OUTD1 zabezpečuje zrušení příznaku blikajícího kurzoru na pozici, do níž bude zapsán nový znak. Při běžném používání displeje pracuje program spolehlivě. Přejdete-li však v programu do strojového kódu, nebo změníte-li v jazyku BASIC pomocí POKE obsah buněk POZICE a RADEK, může se stát, že na obrazovce zůstane „viset" kurzor tam, kde byl před vaším zásahem. Předběhnu a prozradím, že vymazat jej umí (v jazyce BASIC) následující program:

```
10 POKE HEX(3800)+PEEK(HEX(201F))
      *64+PEEK(HEX(2020)),HEX(20)
20 GOTO 20
```

Z této ukázky je také zřejmé, jak pracuje podprogram SESTAV. Adresa 201FH je totiž adresou buňky RADEK, adresa 2020H je adresou buňky POZICE.

Návěštím OUT1 začíná dekódování řídicích znaků. Kódy 0DH, 08H a 0AH jsou kódy pro CR, BS a LF. Je-li některý z nich nalezen, realizuje se patřičná funkce displeje, např. přechod na nový řádek. Není-li nalezen řídicí znak, je vstupní kód upraven na šestibitový kód ASCII. Tento kód je uložen do paměti RAM displeje instrukcí MOV M, A, kurzor se posune na následující místo a po návratu obsahu registrů podprogram končí. Je-li zvolen dvojitý typ písma, je kurzor posunut navíc ještě o jedno místo.

Zdálo by se, že programová obsluha displeje končí. Platí to však pouze pro skupinu 64 grafických symbolů.

Pro volbu typu písma, která je u AND-1 určena obsahem dvou nejvyšších bitů v paměti RAM displeje, je vyhrazena buň-

ka RAM s označením MODE. Před výpisem každého znaku musí tato buňka obsahovat údaj 00 pro běžný znak, 40 pro blikající znak, 80 pro znak, pod nímž bliká kurzor a C0H pro znak dvojnásobné šířky. Před uložením kódu znaku do paměti RAM displeje je pak obsah buňky MODE logicky sečten s šestibitovým kódem znaku.

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0


17910*                  ;MIKRO BASIC JPR-1
17920                   ;*********************************
17930                   ;
17940                   ;   DISPLEJ    JPR-1
17950                   ;
17960                   ;*********************************
17970*                  ;
17980*                  ;ZRUSENI UKAZATELE
17990                   ;
18000 0C06 F5       OUTDIS: PUSH  PSW
18010 0C07 E5               PUSH  H
18020 0C08 D5               PUSH  D
18030 0C09 C5               PUSH  B
18040 0C0A 4F               MOV   C,A
18050 0C0B CDD90C           CALL  SESTAV
18060 0C0E 3A2120           LDA   MODE
18070 0C11 47               MOV   B,A
18080 0C12 7E               MOV   A,M
18090 0C13 E6C0             ANI   0C0H
18100 0C15 FE80             CPI   80H
18110 0C17 C2230C           JNZ   OUTD1
18120 0C1A 78               MOV   A,B
18130 0C1B B7               ORA   A
18140 0C1C C2230C           JNZ   OUTD1
18150 0C1F 7E               MOV   A,M
18160 0C20 E67F             ANI   7FH
18170 0C22 77               MOV   M,A
18180*                  ;
18190*                  ;CR, LF, BS A ULOZENI ZNAKU
18200*                  ;
18210 0C23 79       OUTD1:  MOV   A,C
18220 0C24 FE0D             CPI   0DH
18230 0C26 CAA00C           JZ    CRDS
18240 0C29 FE08             CPI   08H
18250 0C2B CA4B0C           JZ    BACK
18260 0C2E FE0A             CPI   0AH
18270 0C30 CAA70C           JZ    LFDS
18280 0C33 E63F             ANI   3FH
18290 0C35 5F               MOV   E,A
18300 0C36 78               MOV   A,B
18310 0C37 B3               ORA   E
18320 0C38 77               MOV   M,A
18330 0C39 E6C0             ANI   0C0H
18340 0C3D FEC0             CPI   0C0H
18350 0C3D CC5A0C           CZ    IPOZ
18360 0C40 CD5A0C           CALL  IPOZ
18370*                  ;
18380*                  ;NAVRAT
18390*                  ;
18400 0C43 CD930C    OUTD10: CALL  CURRST
18410 0C46 C1               POP   B
18420 0C47 D1               POP   D
18430 0C48 E1               POP   H
18440 0C49 F1               POP   PSW
18450 0C4A C9               RET
18460                   ;
18470                   ;CURSOR ZPET
18480                   ;
18490 0C4B 3A2020    BACK:  LDA   POZICE


ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0


18500 0C4E 3D               DCR   A
18510 0C4F FEFF             CPI   0FFH
18520 0C51 CA430C           JZ    OUTD10
18530 0C54 322020           STA   POZICE
18540 0C57 C3430C           JMP   OUTD10
18550                   ;
18560*                  ;INKREMENT POZICE NA RADKU
18570                   ;
18580 0C5A 3A2020    IPOZ:  LDA   POZICE
18590 0C5D 3C               INR   A
18600 0C5E 322020           STA   POZICE
18610 0C61 FE28             CPI   40
18620 0C63 D8               RC
18630 0C64 AF               XRA   A
18640 0C65 322020           STA   POZICE
18650                   ;
18660                   ;INKREMENT RADKU
18670                   ;
18680 0C68 3A1F20    IRAD:  LDA   RADEK
18690 0C6B 3C               INR   A
18700 0C6C FE18             CPI   24
18710 0C6E CAAD0C           JZ    ROLDIS
18720 0C71 321F20           STA   RADEK
18730 0C74 C9               RET
18740                   ;
18750                   ;NULOVANI DISPLEJE
18760                   ;
18770 0C75 F5       NULDIS: PUSH  PSW
18780 0C76 E5               PUSH  H
18790 0C77 D5               PUSH  D
18800 0C78 C5               PUSH  B
18810 0C79 210038           LXI   H,3800H
18820 0C7C 110008           LXI   D,2048
18830 0C7F CD880C           CALL  NULD10
18840 0C82 CDF00C           CALL  HOME
18850 0C85 C3430C           JMP   OUTD10
18860                   ;
18870 0C88 0E20      NULD10: MVI   C,20H
18880 0C8A 71        NULD20: MOV   M,C
18890 0C8B 23               INX   H
18900 0C8C 1B               DCX   D
18910 0C8D 7A               MOV   A,D
18920 0C8E B3               ORA   E
18930 0C8F C28A0C           JNZ   NULD20
18940 0C92 C9               RET
18950*                  ;
18960*                  ;OBNOVENI UKAZATELE
18970                   ;
18980 0C93 3A2120    CURRST: LDA   MODE
18990 0C96 B7               ORA   A
19000 0C97 C0               RNZ
19010 0C98 CDD90C           CALL  SESTAV
19020 0C9B 7E               MOV   A,M
19030 0C9C F680             ORI   80H
19040 0C9E 77               MOV   M,A
19050 0C9F C9               RET
19060*                  ;
19070*                  ;CR
19080                   ;
19090 0CA0 AF        CRDS:  XRA   A


ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0


19100 0CA1 322020           STA   POZICE
19110 0CA4 C3430C           JMP   OUTD10
19120*                  ;
19130*                  ;LF
19140                   ;
19150 0CA7 CD680C    LFDS:  CALL  IRAD
19160 0CAA C3430C           JMP   OUTD10
19170                   ;
19180                   ;ROLOVANI DISPLEJE
19190                   ;
19200 0CAD 210038    ROLDIS: LXI   H,3800H
19210 0CB0 114038           LXI   D,3840H
19220 0CB3 0E17             MVI   C,23
19230 0CB5 CDCE0C    MOVS:  CALL  MOVR
19240 0CB8 C5               PUSH  B
19250 0CB9 011800           LXI   B,18H
19260 0CBC 09               DAD   B
19270 0CBD EB               XCHG
19280 0CBE 09               DAD   B
19290 0CBF EB               XCHG
19300 0CC0 C1               POP   B
19310 0CC1 0D               DCR   C
19320 0CC2 C2B50C           JNZ   MOVS
19330 0CC5 21C03D           LXI   H,3DC0H
19340 0CC8 114000           LXI   D,64
19350 0CCB C3880C           JMP   NULD10
19360 0CCE 0628      MOVR:  MVI   B,28H
19370 0CD0 1A        MOVZ:  LDAX  D
19380 0CD1 77               MOV   M,A
19390 0CD2 13               INX   D
19400 0CD3 23               INX   H
19410 0CD4 05               DCR   B
19420 0CD5 C2D00C           JNZ   MOVZ
19430 0CD8 C9               RET
19440                   ;
19450                   ;SESTAVENI ADRESY CURSORU DO H,L
19460                   ;
19470 0CD9 2600      SESTAV: MVI   H,0
19480 0CDB 3A1F20           LDA   RADEK
19490 0CDE 6F               MOV   L,A
19500 0CDF 0606             MVI   B,6
19510 0CE1 29        SES10: DAD   H
19520 0CE2 05               DCR   B
19530 0CE3 C2E10C           JNZ   SES10
19540 0CE6 3A2020           LDA   POZICE
19550 0CE9 B5               ORA   L
19560 0CEA 6F               MOV   L,A
19570 0CEB 7C               MOV   A,H
19580 0CEC F638             ORI   38H
19590 0CEE 67               MOV   H,A
19600 0CEF C9               RET
19610                   ;
19620                   ;HOME
19630                   ;
19640 0CF0 AF        HOME:  XRA   A
19650 0CF1 321F20           STA   RADEK
19660 0CF4 322020           STA   POZICE
19670 0CF7 C9               RET
19680 0CF8                  EJECT
```

Nulování displeje zabezpečuje podprogram NULDIS jednoduše tím, že zaplní celou zdánlivou paměť RAM o délce 2K mezerami. Výsledkem je sice to, že do některých míst je mezera uložena dvakrát, programátorovi to však zjednoduší práci.

Hodnoty buněk udávajících pozici a číslo řádku inkrementují rutiny IPOS a IRAD. Kontroluje se, zda jsme na konci řádku (v tom případě číslo pozice vynulujeme a zvětšíme číslo řádku), nebo zda končí poslední řádek (v tom případě provedeme rolování displeje).

Rolování displeje je zabezpečeno postupným přesunem bloků dat pro jednotlivé řádky v paměti RAM displeje a zaplněním posledního řádku mezerami.

Řídicí znak CR způsobí pouze vynulování buňky POZICE, znak LF inkrementaci buňky RADEK a znak BS dekrementaci buňky POZICE.

Poslední rutina, HOME, vynuluje jak buňku RADEK, tak buňku POZICE. Kurzor se tak nastaví do základní pozice.

V programovém řešení obsluhy displeje by bylo pochopitelně možno pokračovat. Komfortnější displeje mají řadu dalších funkcí, umožňujících snadnou editaci textu na obrazovce, vyslání zprávy, tisk obsahu obrazovky pomocí tiskárny atd. Také repertoár řídicích znaků je širší nebo úplný podle normy ASCII.

Pro náš účel by však nemělo smysl zbytečně plýtvat pamětí. AND-1 byl navrhován jako jednoduchý displej, popsané programové vybavení jej zařazuje spíše mezi ty střední; zařazení do vyšší třídy vyžaduje stejně 64 až 80 znaků na řádek. Buďme proto spokojeni a napišme si krátký program:

```
            ORG 2300H

TEST:   CALL CI
        CALL OUTDIS
        JMP  TEST

        END
```

S pomocí monitoru JPR-1 to bude vypadat takto:

```
READY
>MONITOR

MONITOR
*D2300

2300 CD F8 0C
:M:CD F8 0C CD 06 0C C3 00 23
2309 02          : :.
MONITOR
*G2300
```

Posledním příkazem jsme program spustili a můžeme se přesvědčit, že pracuje. Podprogram CI zabezpečí vstup znaku do registru A, podprogram OUTDIS jej vypíše na displeji. Je-li vše v pořádku, máme první krok za sebou. Počítač přestal být negramotným zařízením, umí číst a psát, rozumí nám.

MIKRO BASIC JPR-1

Každý programovací jazyk má své nej . . . U jazyka BASIC to bude asi přívlastek nejpopulárnější. Popularita jazyka BASIC způsobila, že se počet jeho variant počítá již na desítky a že existuje široký výběr i co do výkonnosti a požadavků na použitý počítač.

BASIC vznikl v roce 1963, když se poprvé objevila možnost interaktivně spolupracovat s počítačem. Jako základní komunikační prostředek byl původně používán dálnopis a řada verzí BASIC nese více či méně stop z této doby.

Původně byl BASIC zásadně jazyk interpretační a překladačové verze jsou řešením rozporu mezi jeho velkou popularitou a jeho malou rychlostí.

Je zajímavé, že téměř všechny výhody i nevýhody jazyka BASIC plynou právě z jeho interpretačního charakteru. BASIC umožňuje velmi rychle psát programy, snadno je opravovat, upravovat a téměř ideální je k odlaďování programů, čili k hledání chyb.

Je to proto, že vlastní zdrojový text programu neprochází žádnou fází překladu do strojového kódu. Jednotlivé programové řádky, označené pořadovým číslem, jsou uschovávány v nezměněné podobě, nebo jsou klíčová slova nahrazena jednobytovým kódem. Teprve povelem RUN je spuštěn vlastní interpreter, který postupně prohlíží jednotlivé programové řádky a nalezené programy realizuje.

Vzhledem k tomu, že nezáleží na pořadí, ve kterém řádky píšeme, ale na jejich pořadových číslech, můžeme kdykoli vložit nové řádky mezi staré, přemístit řádky, opravit je nebo vymazat. Při každém novém příkaze RUN se realizuje právě existující verze programu a ladění programu je vlastně neustále opakující se sled úprav textu a ověřování činnosti programu.

Je pochopitelné, že pro efektivní práci musí programátor zvládnout řadu úkonů, jako například používání příkazu STOP, spouštění programu od určitého řádku, zařazování pomocných výpisů, např. hodnot proměnných atd.

Má-li programátor navíc k dispozici tiskárnu, je programování v jazyku BASIC téměř hrou. Většina začátečníků se diví, proč existuje řada dalších vyšších jazyků. Teprve postupně, pomine-li počáteční okouzlení příkazem PRINT nebo FOR TO NEXT, začneme zjišťovat, že i BASIC má své nevýhody. Na většinu nevýhod přijdeme teprve při určité kombinaci našich požadavků a za ně vlastně můžeme sami.

Při opisování cizích programů přijdeme například na to, že BASIC zjišťuje chyby v programu teprve po příkazu RUN. Můžeme proto opsat desítky programových řádků s příkazem, který náš BASIC vůbec nezná, a dozvíme se to, až když interpreter odmítne první z těchto příkazů. Je to proto, že interpreter prohlíží text až po spuštění programu a při psaní textu programových řádků se spokojí s jakoukoli hloupostí.

Snad nejvážnější nectností jazyka BASIC je relativně malá rychlost realizace programu. Při srovnávání bude pochopitelně záležet na druhu použitých příkazů, odhadem však lze říci, že BASIC je 10 až 50krát pomalejší než vlastní strojový kód mikropočítače. A opět je to proto, že při realizaci programu v jazyku BASIC se znovu a znovu prohlíží úplný text programu, jednotlivá klíčová slova se dekódují a teprve potom jsou vyvolávány jednotlivé standardní rutiny napsané ve strojovém kódu.

Typickým příkladem, kdy velmi brzy přijdete na pomalost jazyka BASIC, jsou hry. Fotbal napsaný v BASIC by byl fotbal s lýmy místo a letecká bitva hon na motýly. V takových případech nezbývá nic jiného než sáhnout po strojovém kódu a napsat program např. pomocí textového editoru a překladače assembleru. Bez těchto pomocníků totiž program pro fotbal ani leteckou bitvu jen tak nenapíšete, protože délka programu bude odhadem 2K až 4K bytů.

Další nevýhodou, je-li např. malý rozsah paměti RAM, je značná „upovídanost" jazyka BASIC. Programový text musí být při realizaci programu celý přítomen v paměti počítače a počet volných buněk paměti při psaní programu rychle ubývá.

Poslední zklamání vám BASIC připraví, budete-li potřebovat realizovat jednoúčelové zařízení, např. programátor ústředního topení. Program v BASIC pro tuto aplikaci může mít délku textu okolo 1K bytů. Pro jeho trvalý provoz však musí být v zařízení přítomen také interpreter BASIC o délce např. 4K bytů. A to při ceně pamětí EPROM přijde určitě líto.

Není BASIC jako BASIC

Právě jsme si názorně ukázali jedno ze základních pravidel platících při programování počítačů: za všechny výhody musíme něčím zaplatit.

Neplatí to jen při srovnávání jazyka BASIC s jinými jazyky, ale také při srovnávání různých verzí jazyka BASIC.

Čím výkonnější a komfortnější je BASIC, tím větší paměť potřebuje jeho interpreter a tím pomalejší je realizace většiny výpočtů. Úmyslně zdůrazňuji, že klesající rychlost se týká zejména výpočtů. Je totiž zřejmé, že násobení čísel v rozsahu odpovídajícím 16bitovému číslu bude rychlejší než násobení čísel s přesností na 12 míst, práce s textovými řetězci však u takto rozdílných interpreterů může probíhat stejně rychle.

Zhruba lze říci, že se postupem času vyčlenily čtyři kategorie interpreterů jazyka BASIC. Do první z nich patří celá řada tzv. „tiny" (drobný) BASIC. Interpretery této kategorie mají délku od 3K do 4K bytů a vyznačují se především omezeným rozsahem čísel, omezeným vybavením v oblasti funkcí a minimálními možnostmi práce s textovými řetězci. Podle rozsahu čísel, s nimiž mohou pracovat, mají tyto interpretery často přívlastek „Integer" (celočíselný). Do této kategorie patří také náš Mikro BASIC.

Druhou kategorii, vyhovující nejširšímu okruhu uživatelů, je možno nazvat kategorií standardních verzí BASIC. Pracují s rozsahem čísel odpovídajících přesnosti na 6 platných míst s exponentem např. ±40. Mají již logické operátory AND a OR, soubor goniometrických funkcí a celou řadu příkazů pro práci s textovými řetězci. Délka těchto interpreterů bývá 6K až 8K bytů. Právě tato kategorie proslavila BASIC v prvních deseti letech jeho života.

Třetí kategorie, vhodná zejména pro vědeckotechnické výpočty, se vyznačuje velkou přesností a značným komfortem v oblasti editace programového textu. Interpretery, které mají délku 12K až 16K bytů, pracují s přesností čísel až na 16 platných míst a mívají, v závislosti na typu mikropočítače, také celou řadu nebývalých příkazů v oblasti grafického výstupu na displeji. Do této kategorie je také možno zařadit některé speciální verze BASIC, např. pro řízení reálných procesů.

Poslední, čtvrtá kategorie verzí jazyka BASIC, je zatím mimo možnosti mikropočítače JPR-1. Je to kategorie „diskových basiců", zahrnujících celou řadu příkazů pro práci s datovými a programovými soubory na pružném disku. Délka těchto interpreterů je 16K až 32K bytů s tím, že řada příkazů je uložena na disku a do paměti RAM si je operační systém přesunuje jen v případě potřeby.

Mikro BASIC JPR-1

Mikro BASIC je implementací známého Tiny BASIC pro počítač JPR-1. Vlastní interpreter potřebuje méně než 3K bytů paměti a může být uložen v paměti ROM. Přes všechna omezení je to BASIC velmi příjemný, okouzlující možnostmi spolupráce s programy ve strojovém kódu, hexadecimálním zápisem čísel a řadou dalších překvapení.

I když by bylo možné Mikro BASIC vylepšit, např. doplnit některé funkce, vrátil jsem se po vyzkoušení těchto možností k té jednodušší verzi a doporučuji v případě potřeby sáhnout k některé vyšší kategorii jazyka BASIC. Přísloví o jednoduchosti a kráse zde platí naprosto přesně.

Minimální požadavky na paměť EPROM a jednoduchost Mikro BASIC přinášejí řadu omezení a bude nejlepší začít jejich výčtem.

Aritmetika

Aritmetika pracuje s rozsahem čísel od −32767 do +32767. Odpovídá to rozsahu 16bitových dvojitých registrů CPU 8080 s tím, že nejvyšší 15. bit je vyhrazen pro znaménko čísla. Je-li tento bit 0, je číslo ve zbývajících 15 bitech kladné, je-li 1, je číslo záporné. Začátečníky snad překvapí číslo 15 u nejvyššího, nejvýznamnějšího bitu. Je to proto, že bity počítáme od nuly a to zprava, kde je v registrech nejméně významný bit. Ale zpátky k aritmetice, která pracuje pouze se čtyřmi operátory: +, −, * a /. Logické operátory, např. AND a OR chybí, stejně jako trigonometrické funkce.

Pro názornost jsem připravil dva krátké programy v Mikro BASIC, generující tabulky pro převod čísel mezi dekadickým a hexadecimálním vyjádřením. Úloha 15. bitu je zde zcela zřejmá.

```
 10 FOR I=0 TO -30720 STEP -2048
 20 GOSUB 300
 30 NEXT I
 40 I=-32767
 50 GOSUB 300
 60 END
300 LPRINT I,
310 HARD
320 TAB(5)
330 WORD(I)
335 LPRINT
350 RETURN

READY
>RUN
     0      0000
 -2048      F800
 -4096      F000
 -6144      E800
 -8192      E000
-10240      D800
-12288      D000
-14336      C800
-16384      C000
-18432      B800
-20480      B000
-22528      A800
-24576      A000
-26624      9800
-28672      9000
-30720      8800
-32767      8001
```

```
10 FOR I=0 TO 32767 STEP 2048
20 GOSUB 300
30 NEXT I
40 I=I+2047
50 GOSUB 300
60 END
300 LPRINT I,
310 HARD
320 TAB(5)
330 WORD(I)
335 LPRINT
350 RETURN

READY
>RUN
     0      0000
  2048      0800
  4096      1000
  6144      1800
  8192      2000
 10240      2800
 12288      3000
 14336      3800
 16384      4000
 18432      4800
 20480      5000
 22528      5800
 24576      6000
 26624      6800
 28672      7000
 30720      7800
 32767      7FFF
```

Relační operátory

Relační operátory, neboli operátory pro porovnání dvou čísel poslouží zejména při větvení programů. Celkem máme k dispozici 6 operátorů: >, <, =, #, >= a <=. Pro začátečníky raději názvy: je větší, je menší, rovná se, nerovná se, je větší nebo se rovná a je menší nebo se rovná. Poslední dva operátory nesnesou přehození znaků a operátoru => BASIC nerozumí.

Interpreter pracuje s relačními operátory velice jednoduše. Je-li splněna podmínka daná relačním operátorem, nahradí interpreter celý výraz hodnotou 1, není-li splněna, hodnotou 0. Příslušný příkaz pro větvení programu pak tuto hodnotu zpracuje. Přitom za pravdivý výsledek považuje nejen číslo 1, ale libovolné kladné číslo. Výsledky porovnání je tedy možno sčítat a násobit a nahrazovat tak logické operátory AND a OR. Zatím však alespoň příklad na prezentaci výsledků porovnání.

```
 10 INPUT A
 20 LPRINT A>1,A=2,A<3,A#4

READY
>RUN
A? 1
       0       0       1       1

READY
>RUN
A? 2
       1       1       1       1

READY
>RUN
A? 3
       1       0       0       1

READY
>RUN
A? 4
       1       0       0       0

READY
>RUN
A? 5
       1       0       0       1
```

Jednoduché proměnné

Mikro BASIC dovoluje použít v programu celkem 26 jednoduchých proměnných. „Jméno" jednoduché proměnné je tvořeno vždy jedním písmenem abecedy, tedy A až Z. Hodnota každé proměnné může být v rozsahu −32767 až +32767 stejně jako všechna čísla v Mikro BASIC. Proměnné v Mikro BASIC existují vždy, i když jste je v programu vůbec nepoužili. To proto, že každá z nich má v paměti RAM své pevné místo, odpovídající 2 bytům. Proto můžete napsat PRINT A kdykoli a interpreter vypíše náhodně nastavenou hodnotu z místa uložení proměnné A. Při inicializaci jazyka BASIC se hodnoty proměnných nenulují – pro tento účel slouží příkaz CLEAR.

```
 10 A=2,B=3,Z=-345
 20 LPRINT A,B,Z
 30 CLEAR
 40 LPRINT A,B,Z

READY
>RUN
       2       3    -345
       0       0       0
```

Indexovaná proměnná

Indexovanou proměnnou má Mikro BASIC pouze jednu, označenou znakem, kterému i nejvyšší počítačoví odborníci říkají česky „zavináč": @ . Index v závorce za tímto znakem, v rozsahu 0 až 32767, udává pořadové číslo prvku v poli této proměnné. Každý prvek tohoto pole může mít stejný rozsah hodnot jako jednoduchá proměnná a celé pole leží v paměti RAM počínaje nejvyšší adresou volné paměti RAM, kterou interpreter vyhradil pro text programu. Jednotlivé prvky indexované proměnné postupují tedy zeshora, proti programovému textu. Jinými slovy: počet možných prvků v poli indexované proměnné @ ubývá s přibývajícím programem. Interpreter pochopitelně hlídá okamžik vzájemného střetu a ohlásí jej výpisem SORRY. Příklad na použití indexované proměnné si ukážeme později.

Textový řetězec a textový operátor

Text uzavřený v příkazech PRINT, LPRINT a INPUT v uvozovkách je běžným textovým řetězcem. Tento řetězec si program pouze pamatuje a použije jej v nezměněné podobě. Končí-li příkaz zároveň s programovým řádkem a druhá uvozovka by byla poslední, nemusíte ji psát. Pro interpreter je koncem řetězce druhá uvozovka stejně jako znak CR.

Na rozdíl od textového řetězce je tzv. textový operátor přiřazení dekadické hodnoty kódu ASCII znaku, který je uzavřen mezi dvěma apostrofy. Příkladů na použití textových řetězců bude v dalším textu celá řada a tak si alespoň všimněme textového operátoru.

```
 10 FOR I='A' TO 'B'
 20 LPRINT I,
 30 NEXT I
     65      66
READY
>
```

Základní pravidla Mikro BASIC

Základní jednotkou programu je příkaz. Příkazů může být na řádku i více, oddělíme-li je dvojtečkou.

Vedle příkazů má Mikro BASIC také povely, jejichž funkci si vysvětlíme za chvíli. Povely nebo příkazy očekává BASIC po vypsání vodicího znaku >. Začínáli napsaný text číslem, považuje jej interpreter za programový řádek a zařadí jej do

programového bufferu. Programové řádky musí začínat číslem v rozsahu 1 až 32767. Jestliže řádek s napsaným číslem již existuje, nový řádek jej jednoduše přepíše. Napíšeme-li pouze číslo již existujícího řádku, bude tento programový řádek vymazán.

Jak již bylo řečeno, zapíše interpreter do textového bufferu jakoukoli hloupost začínající číslem. Správnost příkazů prověřuje teprve po povelu RUN.

Dobrým pravidlem je psát programy s číslováním řádků po deseti. Mikro BASIC nemá povel pro přečíslování řádků a číslování po deseti nám umožní vložit dodatečně až 9 nových řádků mezi dva stávající. Dále si můžete zvyknout psát podprogramy s čísly většími např. než 1000, protože obvykle nevíte, jak daleko se v číslování řádků dostanete.

Povely Mikro BASIC

Nenalezne-li interpreter po ukončení řádku znakem CR na jeho počátku číslo, považuje jej za řádek obsahující povel nebo příkazy, které má okamžitě provést. Na rozdíl od příkazů, které mohou být součástí programových řádků, má BASIC několik příkazů, určených pouze pro ovládání interpreteru. Těmto příkazům říkáme povely.

Povely nesmí být v řádku, začínajícím číslem a mohou jim předcházet pouze mezery. Povel ukončíme jako obvykle znakem CR. Až do tohoto okamžiku můžeme, stejně jako při psaní programového řádku, opravovat napsaný text pomocí klávesy BS. Protože první klávesnice mého JPR-1 tuto klávesu neměla, používá místo ní interpreter klávesu LF, která není pro ovládání interpreteru nutná. V tabulce kódů ASCII je proto u klávesy LF hexadecimální kód 08 místo 0A. Klávesou BS vymažeme poslední napsaný znak a kurzor se vrátí o jedno místo vlevo. Jsme-li již na začátku řádku, požádá nás interpreter o psaní nového textu výpisem vodicího znaku na dalším řádku.

Celý řádek můžeme kdykoli před použitím znaku CR zrušit klávesou DEL, čili klávesami „shift" a P. Vraťme se však k povelům Mikro BASIC.

LIST, LLIST

Tento povel slouží v základním tvaru pro výpis programového textu na displeji, ve tvaru LLIST navíc pomocí tiskárny. Protože se tento povel hojně používá, má pro něj Mikro BASIC zkratku, kterou je obyčejná tečka.

Jako parametr příkazu je možno zadat číslo programového řádku, kterým má výpis začít. Napíšeme-li navíc za tímto parametrem znaménko +, můžeme zadat, kolik následujících řádků po zadaném řádku se má vypsat.

Tuto neobvyklou kombinaci jsem u Mikro BASIC zvolil nejen proto, že v žádném jazyce BASIC neexistuje. Nejčastěji používaný způsob se zadáním intervalu řádků, od – do, vyžaduje znát obě čísla a navíc odhadnout, nejsou-li mezi nimi vložené řádky. Příliš mnoho řádků při rolování displeje „uteče" a je lepší zadat za znaménkem + např. 20. Posuďte sami.

RUN

Povel RUN zahájí vlastní realizaci programu počínaje programovým řádkem s nejnižším číslem. Interpreter se vrátí do stavu READY, najde-li příkaz STOP nebo END, narazí-li na chybný příkaz, překročí-li hodnota v aritmetických výrazech povolený rozsah, nebo přerušíme-li program tlačítkem INT.

NEW

Povelem NEW vymažeme všechny existující programové řádky a připravíme tak programový buffer pro psaní nového programu. Nemyslete si však, že interpreter skutečně vynuluje programový buffer v paměti RAM. Zdánlivé vymazání textu je způsobeno pouhým přepisem hodnoty proměnné TXTUNF, kterou používá interpreter pro úschovu první volné adresy paměti RAM. Povelem NEW je do této buňky zapsána první adresa textového bufferu.

Tohoto poznatku často využijete při „oživení" programu, který se vám podařilo např. nechtěně vymazat tlačítkem „reset" na klávesnici. K tomu postačí prohlédnout v monitoru paměť RAM, najít konec programu a zapamatovat si první adresu po posledním znaku CR (hexadecimálně 0D). Potom již jen přepíšeme obsah dvou paměťových míst proměnné TXTUNF, podle výpisu je to adresa 2022H pro uložení méně významných bitů a adresa 2023H pro uložení významnějších bitů zapamatované adresy. Po návratu z monitoru, příkazem R, můžeme vymazaný program opět používat.

MONITOR

Již podle názvu je zřejmé, že povelem MONITOR opustíme Mikro BASIC a JPR-1 očekává příkazy v monitoru. Ten umožňuje snadný zápis programů přímo ve strojovém kódu 8080 pomocí hexadecimálního kódu, prohlížení obsahu paměti, opravy a nakonec návrat k jazyku BASIC. Monitor tedy usnadní zejména přípravu těch částí programu, které jsme se rozhodli napsat, např. z důvodu rychlosti, ve strojovém kódu.

Tyto úseky programu pak budeme volat z jazyka BASIC příkazem CALL. Podrobný popis příkazů monitoru si probereme později.

RAM

RAM je nejen zkratka pro označení typu paměti, do níž píšeme text našeho programu, ale také klíčové slovo povelu, kterým můžeme potřebný prostor v této paměti rozšířit. Parametr příkazu udává novou horní hranici RAM. Nemusí to být jen tehdy, nestačí-li paměť pro další programový text, ale také tehdy, ohlásí-li interpreter nedostatek místa pro uložení prvků indexované proměnné.

Interpreter totiž neustále hlídá horní limit paměti RAM, který mu byl vyhrazen, a přeplnění paměti včas ohlásí. Celá akce posunutí horní hranice RAM je však plná úskalí.

Nelze například snížit horní hranici RAM pod hodnotu, která je dána již při překladu zdrojového textu celého interpreteru. Snížit lze pouze vámi zvýšený limit až na původní velikost. V našem případě, kdy Mikro BASIC používá paměť 1K RAM na desce procesoru, nelze zmenšit paměťový prostor pro programový text pod 702 bytů.

Maximální adresa paměti RAM, s níž může Mikro BASIC pracovat, je dána adresou 7FFFH, což je poslední adresa 32K paměťového prostoru. Vzhledem k tomu, že první volná adresa po prostoru zabraném displejem je 4000H, může mít Mikro BASIC k dispozici maximálně 16K paměti. V tom případě však musí být celý interpreter přeložen se zadáním počátku paměti RAM od adresy 4000H.

Poslední připomínka se týká umístění vstupního řádkového bufferu v paměti RAM při použití povelu RAM. Tomuto bufferu vyhradí interpreter automaticky posledních 64 bytů paměti RAM.

LOAD a SAVE

Podle názvů jsou to typické povely pro úschovu programu a jeho opětovné „natažení" do paměti RAM ve spolupráci s vnějším paměťovým zařízením. Vzhledem k tomu, že při psaní interpreteru nebyl k JPR-1 připojen ještě kazetový magnetofon, jsou podprogramy těchto povelů napsány pro snímač a děrovač děrné pásky. Tyto periférie byly připojeny pomocí čtyř obvodů MH3212 a programová obsluha je zřejmá z výpisu programu.

Po zadání příkazu SAVE se interpreter zeptá na jméno programu a spolu s tímto jménem vyděruje obsah programového bufferu až do konce programového textu. Po příkazu LOAD naopak oznámí jméno programu, vyděrované na děrné pásce, a naplní programový buffer textem programu z této pásky.

Dříve, než si stačíte JPR-1 postavit, a určitě dříve, než si jej budete moci koupit, nahradí jmenované podprogramy jiné, spolupracující s kazetovým magnetofonem s rychlostí přenosu dat 240 zn/s.

Výpisy Mikro BASIC

READY, HOW, WHAT a SORRY již nejsou povely jazyka Mikro BASIC, ale celá jeho slovní zásoba, pomocí které s vámi bude komunikovat. Moc toho není, ale jistě se domluvíte.

WHAT vás bude upozorňovat, že počítač něčemu nerozumí a HOW vás upozorní na překročení číselného rozsahu. Výpisem SORRY vám diplomaticky oznámí, že nemá dostatek paměti RAM buď pro program, nebo pro indexovanou proměnnou, a konečně po READY bude očekávat vaše další příkazy. Podívejte se sami, jak taková komunikace s nepozorným programátorem vypadá.

```
READY
>PRINT"AHOJ"
WHAT?

READY
>LET A=98000
HOW?

READY
>LET @(12345)=234
SORRY
```

Programové příkazy

Pro programové příkazy je typické, že jejich názvy musí být na začátku příkazu, tedy ihned po číslu programového řádku nebo po dvojtečce, oddělující dva příkazy na řádku. Malou výjimkou je příkaz LET.

Programové příkazy jsou výkonnými prvky jazyka BASIC, vyvolávající příslušnou akci. Na rozdíl od nich existují v BASIC také funkce, např. ABS nebo RND, které akci nevyvolávají, vrací však programu číselnou hodnotu. Ty naopak nesmí stát na počátku řádku, ale musí být použity uvnitř některého příkazu. Proto musíme napsat PRINT ABS (–3), potřebujeme-li vypsat absolutní hodnotu čísla –3.

Řada příkazů může být také použita přímo, mimo programový řádek. Někdy se

tomu říká přímý mód nebo kalkulátorový mód jazyka BASIC. Některé příkazy, jako např. REM a STOP, nemají v tomto případě smysl. Hojně však použijeme příkazů PRINT, OUT, GOTO, POKE, LET, BYTE, WORD atd.

V přímém módu lze napsat i celý program a můžete tedy soutěžit, kdo napíše delší či lepší. Příkazy lze psát bez mezer až do délky 64 znaků. Malým příkladem vám napovím.

```
READY
>FORI=1TO10:LPRINTI,I*3,I+4:NEXTI
          1         3         5
          2         6         6
          3         9         7
          4        12         8
          5        15         9
          6        18        10
          7        21        11
          8        24        12
          9        27        13
         10        30        14
```

Vraťme se však k programovým příkazům. Zaměříme se přitom zejména na specifické vlastnosti jazyka Mikro BASIC, protože předpokládám dostupnost seriálů Základy programování samočinných počítačů a Programování v jazyku BASIC, které vycházely v AR v letech 1979 a 1981.

LET

Příkazem LET přiřazujeme hodnotu některé z proměnných. V jediném příkazu můžeme přiřadit hodnoty několika proměnným nebo prvkům indexované proměnné, oddělíme-li jednotlivá přiřazení čárkou. Slovo LET je nepovinné a můžeme tedy přímo psát jméno proměnné, operátor přiřazení (=) a hodnotu proměnné. Místo hodnoty může stát na pravé straně „cokoli". Pojem „cokoli" bude v textu použit vícekrát a my pod ním budeme rozumět proměnnou, výraz, textový operátor, číselnou hodnotu, pseudoproměnnou nebo funkci. Nejlepší však bude příklad.

```
READY
>LLIST
  10 A=3,B=5
  20 C=A,D=B
  30 E='A'
  40 F=234
  50 G=HEX(C0)
  60 H=34*ABS(-4)
  70 LPRINT A,B,C,D
  80 LPRINT E,F,G,H

READY
>RUN
          3          5          3          5
         65        234        192        136
```

A máme-li v počítači několik programových řádků, zopakujme si na příkladu možnosti příkazu LIST.

```
LLIST 60
  60 H=34*ABS(-4)
  70 LPRINT A,B,C,D
  80 LPRINT E,F,G,H

READY
>LLIST+2
  10 A=3,B=5
  20 C=A,D=B
  30 E='A'

READY
>LLIST 30+3
  30 E='A'
  40 F=234
  50 G=HEX(C0)
  60 H=34*ABS(-4)

READY
>LLIST 30+
  30 E='A'

READY
>
```

Všimněte si, že neudanou hodnotu v druhém příkladu považuje příkaz za řádek s nejnižším pořadovým číslem a v posledním příkladu za nulu.

FOR TO NEXT STEP

Celá skupina příkazů slouží k organizaci cyklů. FOR a NEXT jsou samostatné příkazy, TO a STEP jsou pomocná klíčová slova, která mají v celém systému své přesné místo a funkci. Pro STEP = 1 je použití tohoto slova nepovinné. U příkazu NEXT musí být, na rozdíl od dokonalejších interpreterů, uvedeno označení proměnné. Velkou zábavu způsobí možnost zadat parametry cyklu pomocí výrazů, proměnné atd. Pro ilustraci několik příkladů ihned. Řadu dalších najdete na dalších stránkách.

```
READY
>LLIST
   5 LPRINT"NAPIS DVE PISMENA"
  10 FOR I=INCHAR TO INCHAR STEP ABS(-3)
  20 LPRINT I,
  30 NEXT I

READY
>RUN
NAPIS DVE PISMENA
      65        68        71        74
READY
>

LLIST
  10 FOR I=HEX(F2) TO HEX(F6)
  20 LPRINT I,
  30 NEXT I
     242       243       244       245       246
READY
>
```

```
  10 FOR I=1 TO 8
  20 FOR K=1 TO I
  30 LPRINT"A",
  40 NEXT K
  45 LPRINT
  50 NEXT I
A
AA
AAA
AAAA
AAAAA
AAAAAA
AAAAAAA
AAAAAAAA
```

GOTO

Příkaz GOTO přeruší lineární sekvenci při provádění programových příkazů podle čísel programových řádků. Realizace programu pokračuje řádkem, jehož číslo je součástí příkazu. Místo tohoto čísla může být, jak je v Mikro BASIC zvykem, opět „cokoli". Bude-li např. toto číslo řádku výsledkem vyhodnocení výrazu, změní se funkce nepodmíněného skoku na skok podmíněný. Příkladem může být simulace příkazu ON GO TO z „větších" jazyků BASIC. V případě, že výsledné číslo řádky neexistuje, použije BASIC opět nepříjemné slovo HOW?

```
10 LPRINT"10"
20 GOTO 50
30 LPRINT"30"
40 END
45 LPRINT
50 LPRINT"50"
60 GOTO 30
10
50
30

READY
>LLIST
  10 INPUT A
  20 GOTO A*10
  30 LPRINT"30"
  35 END
  40 LPRINT"40"
  45 END
  50 LPRINT"50"

READY
>RUN
A:5
50

READY
>RUN
A:3
30

READY
>RUN
A:4
40

READY
>RUN
A:7
HOW?

READY
>
```

Příkaz GOTO můžete také úspěšně použít v přímém módu, potřebujete-li spustit program od jiné než nejnižší řádky. Často používanou kombinací bude použití příkazu GOTO po zastavení programu příkazem STOP.

GOSUB, RETURN

jsou dva samostatné příkazy, které bez sebe nemohou existovat. Příkazem GOSUB provedeme téměř totéž jako příkazem GOTO. Podobně jako instrukce CALL ve strojovém kódu 8080 však GOSUB zabezpečí zapamatování tohoto místa v programu, v němž byla instrukce použita. Při nalezení příkazu RETURN ve volaném podprogramu pak pokračuje realizace programu tam, kde byl program přerušen. Úmyslně nepíši nic o čísle programového řádku, protože realizace programu může být přerušena příkazem GOSUB i uprostřed programového řádku s několika příkazy. Program pak po nalezení příkazu RETURN pokračuje v provádění dalšího příkazu na původním řádku.

Při používání podprogramů pozor na konec programu. Program musí skončit buď řádkem s největším číslem, nebo, je-li za koncem programu podprogram, příkazem END nebo STOP. Například tak, jak je to v dalším příkladu.

```
    LLIST
 10 FOR I=1 TO 5
 20 GOSUB 50
 30 NEXT I
 40 END
 50 LPRINT RND(7)
 60 RETURN
```

```
READY
>RUN
          4
          5
          6
          4
          1

READY
>RUN
          5
          4
          2
          7
          4
```

REM

Příkaz REM má význam zejména pro dokumentování programů. Pro 1K paměti RAM je však zbytečným luxusem a jeho funkce je jednoduchá. Nedělá nic a zabírá paměťové místo. Budete-li mít však později „širší" paměť, delší a složitější programy, tiskárnu a děravou hlavu jako já, je nepostradatelný.

INPUT

Příkaz slouží pro zadávání dat v průběhu výpočtu z klávesnice. Jedním příkazem INPUT můžeme zadat hodnoty několika jednoduchých proměnných nebo prvků indexované proměnné. Očekávání vstupu dat oznámí interpreter dvojtečkou. Současně s ní může být vypsán název proměnné, libovolný text, nebo obojí. Vhodná varianta se volí tak, jak je ukázáno v dalším příkladu.

```
READY
>LLIST
  10 INPUT A,G,C
  20 INPUT "DELKA ",D
  30 INPUT "VAHA" L
  40 I=3
  50 INPUT "PRVEK"@(I)
  60 LPRINT A,G,C
  70 LPRINT D,L,@(3)

READY
>RUN
A: 12
G: 34
C: 126
DELKA D: 78
VAHA: 167
PRVEK: 123
        12          34          126
        78          167         123
```

PRINT, LPRINT

Nejjednodušší příkaz, který učaruje každému, kdo s programováním v jazyku BASIC začíná. Za klíčovým slovem PRINT (nebo LPRINT pro tiskárnu) je seznam proměnných, výrazů, textových řetězců nebo čísel, vzájemně oddělených čárkami. Není-li na konci tohoto seznamu čárka, je každý příkaz zakončen vypsáním znaků CR a LF. Samotný příkaz bez seznamu pak způsobí vypsání jen těchto znaků. Je-li příkaz zakončen čárkou za posledním prvkem seznamu, čeká kurzor na displeji (nebo psací hlava tiskárny) na následující pozici za posledním vypsaným znakem.

Příkaz PRINT v Mikro BASIC. nezná středník, běžný u ostatních jazyků BASIC. Kromě čárky je však možno využít tzv. formátovacího znaku, kterým je #.

Základní formát tiskne čísla v pěti sloupcích, což při 40 znacích na displeji znamená, že poslední cifra je vždy na pozici 8, 16, 24 ... 40. Základní „rozteč" čísel je tedy 8 pozic a můžeme ji změnit právě pomocí formátovacího znaku #, za nímž následuje hodnota nebo opět „cokoli". Nová hodnota platí pouze pro zbytek prováděného příkazu PRINT nebo do dalšího formátovacího znaku v seznamu téhož příkazu.

Formátovací pravidla se však týkají pouze výpisu číselných hodnot. Textové řetězce, stejně jako výpisy hexadecimálních čísel v příkazech BYTE a WORD, se tisknou bez mezer,a pro jejich oddělení slouží příkaz TAB. Místo popisu však raději uvedu několik příkladů.

```
  10 LPRINTA,B,C
  20 LPRINT#10,A,B,C

READY
>RUN
        1       -567          2
          1         -567              2

  10 FOR I=2 TO 10
  20 LPRINT#I,1,2,3
  30 NEXT I

READY
>RUN
 1  2  3
  1  2  3
   1  2  3
    1  2  3
     1  2  3
      1  2  3
       1  2  3
        1  2  3
         1  2  3

  10 LPRINT"BUDE",#3,22," HODIN"

READY
>RUN
BUDE 22 HODIN

  10 A=23456
  20 B=A/100
  30 LPRINT B,#2,",",A-B*100

READY
>RUN
      234,56
```

Nutnost použít další speciální znak v příkazech PRINT vyvolalo použití displeje AND-1. Jak již víme, má tento displej čtyři typy písma a bylo by škoda, kdyby Mikro BASIC neumožňoval jejich pohodlnou volbu. Se stejnými pravidly jako u formátovacího znaku slouží k tomuto účelu znak *, následovaný číslicí 0 až 3. Protože i zde může být použito „cokoli", vyhodnocuje interpreter pouze nejnižší dva bity výsledné hodnoty a 4 je totéž jako 0.

Je-li jako číslice za znakem pro nastavení typu písma 0, je nastaven základní typ písma, který je také nastaven vždy na počátku každého příkazu PRINT. Je-li číslice 1, bude následovat výpis blikajícími znaky, 2 znamená výpis s blikajícím kurzorem pod každým znakem a konečně 3 nastaví výpis pomocí znaků s dvojnásobnou šířkou.

Nastavený typ písma platí opět do konce příkazu, nebo do další změny typu písma v témže příkazu.

Při této příležitosti musím upozornit na některé záludnosti příkazu LPRINT. Tento příkaz nebude pracovat s tiskárnou stejně jako s displejem, protože tisková řádka je např. 128 a ne 40 znaků. Při 100krát opakovaném příkazu LPRINT A,se nám k našemu údivu vytiskne všech 100 údajů na stejný řádek. Na rozdíl od displeje neumí každá tiskárna po naplnění jednoho řádku přejít automaticky na další řádek a znaky CR a LF se v příkaze LPRINT, zakončeným čárkou, neobjeví. Proto bude nutné použít vhodný čítač cyklů a po vypsání každého, např. 15. údaje zabezpečit výstup znaků CR a LF programově.

TAB

Příkaz TAB má v závorce parametr, udávající, kolik mezer se má vytisknout. Nic více, nic méně. Samozřejmě se pod pojmem vytisknout rozumí vypsat na displeji, stejně jako příkaz PRINT znamená v překladu tisk. To však jsou pozůstatky „dálnopisné" éry a musíme se s tím smířit.

HARD, DISPL

Dvojice příkazů umožňujících přepnout výstup současně na displej i tiskárnu a návrat do základního módu s výstupem pouze na displej. Přidání těchto příkazů si vynutilo to, že mimo příkazů PRINT a LPRINT by musely existovat dvojice TAB – LTAB, BYTE – LBYTE atd. Mnohem jednodušší je přepnout si v pravý okamžik výstup na tiskárnu příkazem HARD a odpojit tento výstup příkazem DISPL. Přitom pozor, tiskárna se navíc vypne na konci každého příkazu LPRINT, povelu LLIST a při návratu do READY. Použití příkazu HARD je zřejmé z řady dalších příkladů. Stejně tak můžeme použít příkaz HARD jako povel, např. před povelem MONITOR. Získáme tak možnost připojit tiskárnu po celou dobu trvání monitoru.

IF

Příkaz IF je podmíněný příkaz, jímž organizujeme v programu větvení výpočtů. Po klíčovém slově IF následuje dvojice aritmetických výrazů nebo proměnných, mezi nimiž je některý z relačních operátorů. Je-li toto porovnání vyhodnoceno jako pravdivé, tzn. je-li nahrazeno nulou, pokračuje program realizací druhé části příkazu, případně dalších příkazů na témže řádku. Je-li vyhodnoceno jako nepravdivé, pokračuje program následujícím programovým řádkem. Druhá část příkazu IF pokračuje nejčastěji klíčovým slovem GOTO, méně často i jiným příkazem, majícím v této souvislosti smysl.

O tom, že můžeme výsledky porovnání sčítat a násobit, jsme si řekli u relačních operátorů, a proto krátce jen dva příklady.

```
 LLIST
  10 HARD:INPUT A
  20 IF (A>2)*(A<5) LPRINT"AND":GOTO10
  30 LPRINT"NE":GOTO10

READY
>RUN
A: 1
NE
A: 2
NE
A: 3
AND
A: 4
AND
A: 5
NE
A:


 LLIST
  10 HARD:INPUT A,B
  20 IF(A>2)*(A<5)+(B>2)*(B<5)LPRINT"AND":GOTO10
  30 LPRINT"NE":GOTO10

READY
>RUN
A: 1
B: 1
NE
A: 1
B: 4
AND
A: 4
B: 2
AND
A: 5
B: 5
NE
A:
```

END, STOP

END znamená konec programu a tento příkaz musíme použít pouze tehdy, není-li konec programu zároveň koncem programového textu. V tomto případě se program zastaví automaticky (i když END není nikdy na škodu). Příkaz STOP způsobí totéž s tím, že se s hvězdičkou vypíše řádek obsahující příkaz STOP. Je to proto, abychom mohli při ladění programů umístit do programu více příkazů STOP. Po zastavení programu pak víme, jak se program větvil, můžeme vypsat hodnoty proměnných, změnit je a pokračovat příkazem GOTO s udáním čísla řádku.

CLEAR

Příkaz vynuluje všech 26 jednoduchých proměnných. Budeme-li potřebovat nulovat prvky indexované proměnné @, využijeme programového cyklu FOR TO NEXT, nebo vynulujeme paměť RAM v místě uložení prvků indexované proměnné pomocí příkazů POKE.

CLS

CLS je nejjednodušší příkaz, kterým se vynuluje obrazovka displeje a nastaví kurzor na začátek prvního řádku.

CALL

Příkaz CALL skutečně potěší. Doplněný funkcí HEX splňuje sny každého programátora o jednoduché spolupráci programů v BASIC a strojovém kódu mikroprocesoru. Bez jakýchkoli zbytečností zabezpečí skok do programu ve strojovém kódu a po nalezení příslušné instrukce RET pokračuje program v BASIC dále.

Příkaz neumí předat žádné parametry „ani tam, ani zpět". Pomocí příkazu POKE a funkcí PEEK a HEX to však zvládnete velmi snadno. Budete-li navíc hračičkové, naučíte se ve strojovém kódu pracovat nejen s hodnotami jednoduchých proměnných jazyka BASIC, ale i s prvky indexované proměnné.

Jako příklad spojení programu v BASIC s programem ve strojovém kódu je uveden příklad testu stisknutí tlačítka T na klávesnici. Nejprve je uveden zdrojový text programu ve strojovém kódu tak, jak bychom jej napsali v assembleru, dále následuje program v BASIC, příklad vložení strojového prógramu do paměti RAM pomocí Monitoru a konečně ukázka spuštění programu.

```
        ORG 2200H

TLAC:   LDA 2800H
        RLC
        RLC
        RNC
        JMP TLAC
```

```
LLIST
  10 LPRINT"STISKNI TLACITKO"
  20 CALL HEX(2200)
  30 LPRINT"DEKUJI"

READY
>MONITOR

MONITOR
*D2200

2200 23          :#:3A 00 28
2203 F4 56 F2    :t:07 07 D0
2206 56          :V:C3 00 22
2209 44          :D:.

MONITOR
*R
READY
>HARD

READY
>RUN
STISKNI TLACITKO
DEKUJI
```

POKE

Příkaz vzbuzující zpočátku nedůvěru každého programátora. S výslovností „pouk" znamená strčit, uložit, zasunout do paměti. Když uložit, tak musíme říci kam a co? První parametr proto udává adresu v 64K paměťovém prostoru, druhý, oddělený čárkou, údaj, který bude na tuto adresu uložen. Oběma parametry může být opět naše „cokoli". Oba však mají své zvláštnosti. Začnu druhým, který má reálný rozsah 0 až 255, protože se jedná o hodnotu ukládanou do jednoho bytu paměti. Je-li výsledek větší, pracuje interpreter pouze s hodnotou spodních 8 bitů.

S prvním parametrem je to ještě horší. Mikro BASIC pracuje s čísly, která mají 15., nejvyšší bit rovný 1, jako se zápornými. Potřebujeme-li proto uložit příkazem POKE cokoli do horních 32K paměti RAM, například parametr programu ve strojovém kódu, musíme použít záporné desítkové číslo. Většina větších interpreterů BASIC ani jinou možnost nemá a je příjemné, že u Mikro BASIC můžeme tuto složitou operaci obejít hexadecimálním vyjádřením pomocí funkce HEX. Stejná pravidla platí i u příkazů CALL, O$, I$ a funkce PEEK, kde také můžeme adresovat celý prostor paměti 64K.

Následující příklad je při praktickém ověřování příkazu POKE přímo na JPR-1 dostatečně přesvědčivý. Způsobí totiž střídání 1 a 0 na výstupu bitu 6 výstupního portu 0. Vzhledem k tomu, že tento bit ovládá akustickou signalizaci klávesnice, je výsledek zřetelně slyšet.

```
10 POKE HEX(2400),64
20 POKE HEX(2400),0
30 GOTO 10
```

OUT

Tento příkaz oceníte zejména při připojování dalších přídavných zařízení, ovládání nejrůznějších výkonových a signalizačních prvků a testování hardware. První parametr příkazu udává číslo výstupního portu. Druhý parametr, oddělený čárkou, udává hodnotu, která bude do tohoto portu zapsána provedením příkazu.

Reálné hodnoty obou parametrů jsou 0 až 255 a místo nich může stát opět „cokoli". Interpreter rozsah parametrů nehlídá a použije jednoduše spodních 8 bitů.

OUTCHAR

Příkaz provede výstup znaku, jehož dekadická hodnota je uvedena v závorce. Výstup se objeví na displeji a pokud by byl použit příkaz HARD, také na tiskárně. V závorce může být místo hodnoty opět „cokoli". Příkaz slouží především pro výstup nestandardních znaků, které nemohou být součástí textových řetězců, nebo když znaky vznikají jako výsledek výpočtu. Nejprůkaznější je opět příklad.

```
10 FOR I='A' TO 'A'+ 8
15 HARD
20 OUTCHAR (I):TAB(2)
30 LPRINT I,
35 TAB(2)
40 HARD:BYTE(I):LPRINT
50 NEXT I
```

```
READY
>RUN
A          65   41
B          66   42
C          67   43
D          68   44
E          69   45
F          70   46
G          71   47
H          72   48
I          73   49
```

BYTE, WORD

Dva speciální příkazy Mikro BASIC, které nenajdeme u „větších" jazyků BASIC. Parametr v závorce (nebo také „cokoli") představuje dekadickou hodnotu, která bude vytištěna v případě příkazu BYTE jako hexadecimální číslo 00 až FF, v případě WORD jako hexadecimální číslo v rozsahu 0000 až FFFF. Rozsah parametru u příkazu BYTE není opět kontrolován. Za všechny příklady ten nejprůkaznější, který dokáže vypsat hexadecimální obsah paměti JPR-1 od adresy 0 do adresy 0FFFH tak, jak jej vypisují hexadecimálně orientované monitory.

```
 5 FOR I=0 TO HEX(FFF) STEP 256
10 FOR T=0 TO 255 STEP 16
15 HARD
20 WORD(I+T):TAB(2)
30 FOR  K=0 TO 15
40 BYTE (PEEK(I+T+K)):TAB(1)
50 NEXT K
60 LPRINT
70 NEXT T
80 HARD:OUTCHAR(HEX(0C))
90 NEXT I
```

MASK

Tak jako v příkazu IF se podaří obejít chybějící logické operátory AND a OR různými programátorskými triky. Nejcitelněji si však potřebu logické operace AND uvědomíme, potřebujeme-li zpracovat údaj z některého vstupního portu mikropočítače a zajímá-li nás např. hodnota jediného bitu. Jistěže lze pomocí logických posuvů hodnoty, získané pomocí funkce IN, správnou informaci získat. BASIC však příkaz pro posuv nemá a je ho tedy nutné imitovat násobením čísla dvěma nebo použít podprogram ve strojovém kódu.

Proto jsme do Mikro BASIC přidali příkaz MASK, který nastaví vstupní masku pro použití funkce INM. Tato funkce zabezpečí vstup údaje z daného portu a navíc provede logický součin s nastavenou maskou. Počáteční hodnota „masky" před každým spuštěním programu je FF (hexadecimálně). Příkazem MASK ji můžeme změnit tak, že nová hodnota masky tvoří parametr příkazu. Potřebný příklad bude uveden u funkce INM.

WAIT

je běžným příkazem jazyka BASIC pro zadání časového intervalu. Rozsah parametru je 0 až 255 s tím, že si přesnou kalibraci musíte udělat sami. Záleží totiž na kmitočtu použitého krystalu a u mého počítače znamená každá jednotka para-

metru asi 0,1 s. Při interpretaci tohoto příkazu počítač jednoduše čeká a program nelze přerušit tlačítkem INT na klávesnici. To proto, že test na přerušení programu je vždy na konci každého příkazu.

BEEP

Podobný příkaz jako předcházející. Rozdíl je pouze v tom, že počítač nejen čeká, ale také vydává akustický signál. Rozsah parametru je 0 až 32767. Dva společné příklady na příkazy WAIT a BEEP jsou důkazem toho, že jako parametr příkazů může být opět naše „cokoli".

```
10 WAIT RND(8)
20 BEEP RND(250)
30 GOTO 10

10 WAIT 5
20 BEEP 25
30 WAIT 5
40 BEEP 150
50 GOTO 10
```

I$

Dva příkazy pro vstup a výstup textových řetězců jsem si jako lahůdku jazyka Mikro BASIC nechal na konec celé skupiny příkazů. Jsou totiž, společně s funkcí LEN, malou náplastí na omezení práce s textovými řetězci tak, jak ji známe z „větších" BASIC. Příkaz I$ má jeden parametr, znamenající adresu v paměti RAM. Při provádění příkazu program čeká na vstup znaků z klávesnice a interpreter zabezpečuje ukládání jejich kódů od určené adresy. Příkazy se přeruší po ukončení textu znakem CR. Interpreter zapíše do paměti za poslední znak 00 a navíc aktualizuje pseudoproměnnou LEN, o které si povíme později. Pro srozumitelnost příkladu, následujícího za příkazem O$, však prozradím, že LEN má hodnotu, která se rovná počtu znaků zapsaného textového řetězce.

O$

Příkaz pro výstup textového řetězce je ještě jednodušší. Od adresy, zadané parametrem, vypíše text z paměti až po první byte s hodnotou 00. Pro větší srozumitelnost je po následujícím příkladu vypsán obsah paměti RAM pomocí Monitoru.

```
LLIST
  10 I$ HEX(2200)
  20 LPRINT LEN
  25 HARD
  30 O$ HEX(2200)

READY
>RUN
ABCDEF12
       8
ABCDEF12
READY
>HARD

READY
>MONITOR

MONITOR
*D2200
2200 41          :A:
2201 42          :B:
2202 43          :C:
2203 44          :D:
2204 45          :E:
2205 46          :F:
2206 31 32 00    :12:
2207 32 00 44    :2:D
2208 00          :.
MONITOR
*R
READY
>
```

Funkce Mikro BASIC

Jak již bylo řečeno, nemohou stát klíčová slova funkcí na počátku příkazu, ani nemohou být samostatně použita v přímém módu jazyka BASIC. Funkce vrací interpreteru číselnou hodnotu a může tedy stát tam, kde očekáváme číslo. Navíc potřebuje funkce určitou specifikaci, podle které nám vrátí potřebnou hodnotu. U některých funkcí tak jde o převod mezi kódy, transformaci adresy v paměti RAM na obsah této paměťové buňky, anebo překódování informace o stisku klávesy na klávesnici na kód příslušného znaku.

Mikro BASIC nemá funkcí mnoho, začínající programátor si však i tak přijde na své. „Pružnost" jazyka Mikro BASIC je taková, že zadání funkce přijme i jako údaj pro příkaz INPUT. Místo čísla můžeme proto při vstupu dat napsat např. HEX (3CF).

RND

Funkce generuje náhodně zvolené číslo v rozsahu intervalu od 1 do hodnoty, která je uvedena jako parametr v závorce.

ABS

Vrací absolutní hodnotu čísla, které je uvedeno jako parametr v závorce. Jako parametr může být opět uvedeno „cokoli". A pro oživení malý příklad:

```
  10 FOR I=8 TO -12 STEP -3
  20 LPRINT #4,ABS(I),
  30 NEXT I

READY
>RUN
   8    5    2    1    4    7   10
READY
>
```

HEX

Funkce, která opět potěší příznivce programování ve strojovém kódu. Vrací totiž dekadickou hodnotu mezi −32767 a +32767 čísla, které je uvedeno v závorce v hexadecimálním tvaru. Na rozdíl od pravidel zápisu v assembleru se levé nuly nepíší a nepíše se ani označení H za tímto údajem.

INCHAR

Funkce nemá parametr. Parametr je vlastně dodán formou stisknutí klávesy na klávesnici a funkce INCHAR vrátí dekadickou hodnotu příslušného znaku ASCII. Názorný příklad je poučný nejen pro objasnění této funkce, ale znázorňuje rozdílnost výstupů pomocí příkazů PRINT, BYTE a OUTCHAR.

```
LLIST
  10 A=INCHAR
  20 IF A=HEX(0D) END
  25 HARD
  30 OUTCHAR A:TAB(2):LPRINT A,
  35 TAB(2):BYTE(A):LPRINT
  40 GOTO 10

READY
>RUN
1          49   31
9          57   39
A          65   41
K          75   4B
*          42   2A
?          63   3F
!          33   21
5          53   35
Z          90   5A
```

PEEK

Tradiční protipól příkazu POKE. S výslovností „pík" znamená doslova „kouknout se do paměti". Parametr v závorce udává adresu, z níž budeme číst a funkce vrací hodnotu bytu na této adrese. Pro názornost jsem připravil příklad, který nečte z paměti RAM, ale čte hodnotu z paměťově adresovaného portu 1. Celý program řeší v jazyce BASIC stejný problém, jako příklad uvedený u příkazu CALL. Jak vidíte, řešení v jazyce BASIC je mnohem jednodušší.

```
  10 LPRINT"STISKNI TLACITKO"
  20 A=PEEK(HEX(2800))
  40 IF A=191 LPRINT"DEKUJI":END
  50 GOTO 20

READY
>RUN
STISKNI TLACITKO
DEKUJI
```

IN, INM

Protějšek příkazu OUT vrací hodnotu 0 až 255 převzatou ze vstupního portu, jehož číslo je uvedeno jako parametr v závorce. Funkce INM navíc provede logický součin se vstupní maskou, nastavenou příkazem MASK. Nejlépe to pochopíte z následujícího příkladu.

```
  10 LPRINT IN (HEX(FF)),:TAB(2)
  20 LPRINT INM (HEX(FF))
  30 MASK 7
  40 LPRINT IN (HEX(FF)),:TAB(2)
  50 LPRINT INM (HEX(FF))

READY
>RUN.
      255          255
      255            7
```

LEN, TOP, SIZE

Tyto tři funkce nejsou plnohodnotnými funkcemi. Nedostávají totiž žádný viditelný parametr. Spíše jim vyhovuje název pseudoproměnné. Vracejí sice hodnotu, kterou lze pomocí příkazů běžně zpracovat, jejich nastavení však nemá na starosti programátor, ale interpreter.

Pseudoproměnná LEN například udává, kolik znaků vstoupilo do paměti RAM při posledním použití příkazu I$.

Pseudoproměnná TOP vrací zase hodnotu, rovnající se adrese první volné buňky paměti RAM za programovým textem. S rostoucí délkou programu se hodnota TOP zvětšuje a udává, kam můžeme beztrestně zapisovat textové řetězce pomocí příkazů I$, psát programy ve strojovém kódu atd.

Konečně pseudoproměnná SIZE udává, kolik volných bytů v paměti RAM máme ještě k dispozici pro další program a prvky indexované proměnné.

Příklad na použití pseudoproměnné LEN byl již uveden. Příklady na výpis hodnot TOP a SIZE máte před sebou.

```
READY
>NEW

READY
>LPRINT TOP:HARD:WORD(TOP)
     8430
20EE
READY
>10 REM
>LPRINT TOP:HARD:WORD(TOP)
     8436
20F4
READY
>NEW

READY
>LPRINT TOP:HARD:WORD(TOP)
     8430
20EE
READY
>
```

```
  10 LPRINT"PRO PROGRAM MATE K DISPOZICI"
  20 LPRINT"JESTE",@5,@IZE," BYTU PAMETI RAM"

READY
>RUN
PRO PROGRAM MATE K DISPOZICI
JESTE  620 BYTU PAMETI RAM
```

Závěrem několik příkladů

Teprve když známe všechny příkazy a funkce jazyka Mikro BASIC, můžeme se podívat na čtyři příklady. Zejména bude vhodné si ukázat, jak se pracuje s indexovanou proměnnou a textovými řetězci.

První program zjišťuje, zda je číslo z intervalu 1 až 100 prvočíslem. Zajímavé jsou v něm příkazy IF pro větvení programu a kombinované výpisy textu a čísel.

```
LLIST
  5 LPRINT:HARD
 10 INPUT"CISLO"A
 15 IF (A=0)+(A>100) GOTO 10
 20 B=1,I=0
 30 B=B+1
 35 IF B>A/2 GOTO 100
 40 C=A/B
 50 IF A@C:B GOTO 30
 60 LPRINT"CISLO",@5,A," JE DELITELNE",B
 70 I=1
 80 GOTO 30
100 IF I=1 GOTO 5
110 LPRINT"CISLO",@5,A," JE PRVOCISLO"
120 GOTO 5

READY
>RUN

CISLO:4
CISLO    4 JE DELITELNE    2

CISLO:7
CISLO    7 JE PRVOCISLO

CISLO:123
CISLO:0
CISLO:12
CISLO    12 JE DELITELNE    2
CISLO    12 JE DELITELNE    3
CISLO    12 JE DELITELNE    4
CISLO    12 JE DELITELNE    6

CISLO:97
CISLO    97 JE PRVOCISLO

CISLO:2

*  10 INPUT"CISLO"A

READY
>
```

Druhý program je ukázkou použití indexované proměnné. První část programu zajišťuje vstup hodnot jednotlivých prvků indexovaného pole, druhá část výpis těchto hodnot.

```
 HARD

READY
>LLIST
  10 FOR I=0 TO 5
  20 HARD:INPUT @(I)
  30 NEXT I
  40 LPRINT
  50 FOR I=0 TO 5
  60 LPRINT @5,@(I),
  70 NEXT I

READY
>RUN
@(I):12
@(I):23
@(I):-34
@(I):678
@(I):-4
@(I):2

   12   23  -34  678   -4    2
READY
>
```

Třetí program řeší podobný problém s tím rozdílem, že počet prvků může být 100, pokud nemá některý prvek nulovou hodnotu. Při zadání nulové hodnoty je vstup prvků ukončen a hodnoty prvků jsou vypsány podle velikosti. Celý program má tedy tři samostatné části: vstup dat, třídění a výstup dat.

```
READY
>LLIST
  10 FOR I=0 TO 99
  20 HARD:INPUT @(I)
  30 IF @(I)=0 GOTO 60
  40 NEXT I
  60 FOR X=0 TO I-1
  70 FOR Y=0 TO I-X-1
  80 IF @(Y+1)>=@(Y) GOTO 100
  90 Z=@(Y):@(Y)=@(Y+1):@(Y+1)=Z
 100 NEXT Y:NEXT X
 105 FOR K=0 TO I-1
 110 LPRINT @5,@(K),
 120 NEXT K

READY
>RUN
@(I):12
@(I):-6
@(I):23
@(I):56
@(I):8
@(I):-1
@(I):87
@(I):0
  -6   -1    0    8   12   23   56
READY
>
```

Poslední program má také tři relativně samostatné části. První úsek programu zajistí vstup tří textových řetězců do paměti RAM s tím, že počáteční adresy těchto tří textů jsou uloženy ve třech prvcích indexované proměnné. Druhý úsek programu generuje postupně kombinace tří indexů I, K, L a třetí úsek programu převádí každou z kombinací na výpis tří uložených textů v pořadí odpovídajícím této kombinaci. Poslední úsek programu je napsán jako podprogram, i když by mohl být součástí vlastního programu. Všimněme si však jiných podrobností.

Řádek 10 určuje, že očekávané textové řetězce budou ukládány ihned za text programu, tedy od adresy TOP. Hodnota proměnné R je pak po každém vstupu změněna o délku řetězce a navíc je přeskočena koncová nula, označující konec textu. To proto, aby příkaz O$ vypsal pouze jeden text. Jiný způsob práce s textovými řetězci může být ten, že si kromě počáteční adresy textu zapamatujeme také příslušnou hodnotu LEN. K tomu však potřebujeme vždy dva prvky indexované proměnné.

```
READY
>LLIST
  10 R=TOP
  20 FOR T=1 TO 3
  30 @(T)=R
  40 I$ R
  50 R=R+LEN+1
  60 NEXT T
  65 LPRINT
  70 FOR I=1 TO 3
  80 FOR K=1 TO 3
  90 IF K=I GOTO 130
 100 FOR L=1 TO 3
 110 IF (L=I)+(L=K) GOTO 120
 115 GOSUB 300
 120 NEXT L
 130 NEXT K
 140 NEXT I
 150 END
 300 HARD
 305 O$ @(I):TAB(1)
 310 O$ @(K):TAB(1)
 320 O$ @(L)
 330 LPRINT:RETURN

READY
>RUN
DNES
PADA
SNIH

DNES PADA SNIH
DNES SNIH PADA
PADA DNES SNIH
PADA SNIH DNES
SNIH DNES PADA
SNIH PADA DNES
```

Mikromonitor JPR-1

U jednodeskových mikropočítačů a stavebnic bývá monitor jediným programem. Teprve po „natažení" dalšího programu do paměti RAM, nebo po přidání paměti EPROM s programem můžeme pomocí některého příkazu monitoru tento program spustit.

V základním programovém vybavení JPR-1 má Mikromonitor vedlejší funkci. Proto je také po zapnutí počítače nejprve odstartován interpreter jazyka Mikro BASIC a do monitoru můžeme přejít pomocí povelu MONITOR.

Není-li Mikromonitor hlavním programem, musí také zabírat v paměti EPROM co nejméně místa. Proto byla zvolena úsporná varianta, bez zbytečných příkazů, které si můžeme podle potřeby snadno napsat v jazyce BASIC. Přesto je, vzhledem ke své délce, Mikromonitor poměrně efektivním prostředkem pro práci se strojovým kódem.

Příkazy Mikromonitoru

Po přechodu do monitoru povelem MONITOR očekává počítač příkazy vypsáním hvězdičky. Příkazy jsou jednopísmenové a první tři z nich vyžadují zapsat jeden parametr. Než si je postupně probereme, podívejme se na jejich přehled. Zkratka „adr", znamená parametr, udávající hexadecimálně adresu v rozsahu 0 až FFFF.

Dadr výpis obsahu paměti s možností změny
Sadr skok do podprogramu s návratem do monitoru
Gadr odstartování programu
R návrat do jazyka Mikro BASIC bez zrušení programu
B inicializace Mikro BASIC

Adresa se zadává hexadecimálně, levé nuly není nutno psát, za písmenem příkazu nesmí být mezera a při chybě stačí pokračovat správným zadáním parametru, monitor vezme v úvahu pouze poslední čtyři znaky. Chybí-li parametr vůbec, nahradí jej monitor nulou. Všechna nesprávná zadání a použití nedovolených znaků monitor odmítne a čeká na další příkaz.

Na první pohled se zdá seznam příkazů Mikromonitoru velmi chudý. Příkaz D je však velmi univerzálním příkazem a nahrazuje vlastně pět příkazů běžných u monitorů jednodeskových mikropočítačů. Podívejme se tedy na příkazy Mikromonitoru blíže.

Příkaz D

Nejběžnější příkaz všech monitorů (display memory) u JPR-1 zcela postačí pro zápis a opravy programů ve strojovém kódu. Po zadání parametru, kterým je počáteční adresa, a ukončení zadání příkazu znakem CR, se vypíše hexadecimální obsah paměťové buňky a zároveň odpovídající znak podle tabulky ASCII. Navíc nám monitor zjistí délku instrukce, odpovídající tomuto bytu podle instrukčního souboru 8080 a vypíše nám hexadecimálně i druhý nebo třetí byte instrukce. Je to tedy jakýsi náznak reverzního assembleru s tím, že kódy nejsou vypisovány symbolicky. V každém případě je to velmi příjemné při hledání chyb a opravách. Výpis další instrukce pokračuje po každém dalším stisknutí klávesy CR. Na začátku každého řádku je vždy vypsána adresa, zvětšená pochopitelně o délku předcházející instrukce. Není-li byte vypsaný na prvním místě operačním kódem, můžeme přerušit přičítání délky instrukce k počáteční adrese a pokračovat výpisem následujícího bytu bez ohledu na její délku. K tomu stačí použít klávesu SP (mezera).

Aby to nebylo tak jednotvárné, můžeme se klávesou BS (v našem případě LF) vracet k předcházející adrese a „procházet se" tak libovolně po paměti počítače. Nejlépe vám to objasní příklad.

```
READY
>MONITOR

MONITOR
*DO

0000 C3 19 00   :C:
0003 2A 22 20   :*:
0006 23         ---- :#:   } CR
0007 C9         :I:
0008 C3 EE 23   :C:
0009 EE 23      :n:
000A 23         :#:   } SP
000B 2A 01 20   :*:
000E 2B         :+:   -CR
000D 20         : :
000C 01 20 2B   : :
000B 2A 01 20   :*:   } BS
000A 23         :#:
0009 EE 23      :n:
0008 C3 EE 23   :C:
000B 2A 01 20   :*:   } CR
000E 2B         :+:
000F C9         :I:.
MONITOR
*R
READY
>
```

S pouhým procházením po paměti počítače však nevystačíme, proto nám tentýž příklad D umožňuje kdykoliv zadávat nový obsah paměti a to navíc jak v hexadecimálním kódu, tak v kódu ASCII. V druhém případě uzavřeme text do uvozovek. Nové údaje můžeme oddělovat mezerou nebo po každém z nich použít znak CR. Příklad ukončíme zadáním obyčejné tečky. Následující příklad dokumentuje možnosti zápisu nových údajů, pochopitelně pouze do paměti RAM.

Příkazy S a G

Oba příkazy zabezpečí skok do programu na zadané adrese. Příkaz S (subroutine call) navíc zajistí návrat do monitoru po nalezení příslušné instrukce RET. Příkaz G (go) je prostým skokem bez návratu. První příkaz používáme při ladění programů v monitoru, druhý pro spuštění programu ve strojovém kódu.

```
READY
>MONITOR

MONITOR
*D2300

2300 04          : :
2301 43          :C:
2302 20          : :
2303 00          : :
2304 10          : :
2305 20          : :12 23 45 67
2309 02          : :34
230A 60          : :56
230B 00          : :
230A 56          :V:
2309 34          :4:
2308 67          :g:
2307 45          :E:
2306 23          :#:
2305 12          : :
2304 10          : :"ABCDEF"
230A 56          :V:
2309 46          :F:
2308 45          :E:
2307 44          :D:
2306 43          :C:
2305 42          :B:
2304 41          :A:.
MONITOR
*B
```

Příkazy R a B

Oba příkazy zabezpečí návrat do jazyka Mikro BASIC. Příkaz R (return) přitom neudělá vůbec nic a rozepsaný program v jazyce BASIC je v tom stavu, v jakém jsme ho opustili před povelem MONITOR. Příkaz B (BASIC) naproti tomu provede totéž jako tlačítko RESET na klávesnici. Inicializace jazyka BASIC má v tomto případě za následek vymazání celého programu.

Výpis programu

Výpis celého programu ve zdrojovém tvaru má přes čtyřicet stran formátu A4. Bez hlubšího popisu by tento výpis přinesl užitek pouze těm zkušenějším. Proto je celý program otištěn pouze formou hexadecimálního výpisu pamětí EPROM tak, jak po řadu měsíců v počítači JPR-1 pracoval. Některé pozdější (drobné) úpravy nejsou v této verzi zahrnuty. To proto, aby popis programu přesně souhlasil s obsahem paměti EPROM.

Jak znám čtenáře AR, a také podle vlastních zkušeností, najde se řada těch, kteří budou chtít za každou cenu program pochopit nebo alespoň upravit pro jiný mikropočítač. Pro ty uvedu alespoň nejdůležitější adresy a informace. V žádném případě bych nechtěl, aby někdo z vás program překládal pomocí zpětného assembleru a rekonstruoval celý zdrojový text. Výpis programu existuje, není tajný, a bude-li zájem, najde se jistě vhodná příležitost program otisknout na stránkách AR nebo jinak zpřístupnit.

Znovu připomínám, že program o délce 4K je přeložen od adresy 0000H, tedy od počátku paměťového prostoru, může být uložen v pamětech typu ROM a předpokládá umístění paměti RAM od adresy 2000H s minimální kapacitou 1K. Vzhledem k tomu, že předpokládám zájem o Mikro BASIC také u těch, kteří mají jiný mikropočítač než JPR-1, je uveřejněný program určen pro uložení do pamětí 2K typu 2716. Paměťový prostor JPR-1 je totiž při osazení pamětmi 1K typu 2708 „přerušovaný" a každé „kilo" paměti začíná s roztečí 2K. Program musí být v tomto případě přeložen s mezerou 1K po každém úseku 1K programu.

Důležité adresy programu

Interpreter jazyka Mikro BASIC začíná od adresy 0H. Tam je však uložena pouze instrukce pro skok programu na adresu 19H, kde je skutečný, tzv. studený start interpreteru. Při skutečném startu je vynulován programový buffer, rozsah paměti RAM je nastaven na základní hodnotu a interpreter čeká na vaše příkazy.

Tzv. horký start začíná na adrese 4DH. Návrat interpreteru na tuto adresu poznáte jednoduše výpisem READY. Většinou se sem interpreter vrací po chybě, nebo nalezne-li konec programu. Výpis vodicího znaku a čekání na vstup řádku začíná na adrese 70H.

Od adresy DCH začíná tabulka klíčových slov, která bude zajímat zejména zvídavé programátory. Tabulka končí na adrese 261H a pro každý povel, příkaz nebo funkci je v ní uložen text klíčového slova v kódu ASCII, jeden nulový byte a dvoubytová adresa prováděcí rutiny.

Další řada adres bude zajímat ty, kteří budou upravovat vstupy a výstupy programu pro využití na jiném mikropočítači. Všechny rutiny pro vstup a výstup pracují s registrem A a jejich adresy jsou: B68H – děrování znaku, B7FH – vstup znaku ze snímače děrné pásky, BE0H – výstup znaku na displeji a tiskárně současně, C06H – výstup znaku na displeji a konečně CF8H – vstup znaku z klávesnice.

Mnohem důležitější je však informace, kde všude je třeba v programu tyto adresy změnit, potřebujeme-li přizpůsobit celý program pro použití jiných vstupních a výstupních rutin: pro děrovač jsou to adresy B61H, B9DH, BA1H, BAFH a BD0H; pro snímač B23H, B37H, B3FH, B43H a B50H; pro tiskárnu BF4H; pro displej BE1H a C00H a konečně pro klávesnici jsou to adresy 7ADH, ACEH, E0EH, EB2H, F30H, F78H a F9EH.

```
HEXADECIMALNI VYPIS EPROM JPR-1                    STRANA  1

0000   C3 19 00 2A 22 20 23 C9 C3 EE 23 2A 01 20 2B C9
0010   C3 F7 23 CD 63 07 C3 4D 00 31 AF 20 CD FA 0B CD
0020   75 0C AF 32 21 20 32 00 20 11 D0 00 CD 17 08 21
0030   03 00 22 1D 20 21 ED 20 22 22 20 21 AD 23 22 E9
0040   20 21 ED 23 22 EB 20 21 AB 23 22 E7 20 CD F2 08
0050   31 AF 20 11 F5 02 97 32 21 20 2F 32 0A 20 CD 17
0060   08 21 68 00 22 0B 20 21 00 00 22 13 20 22 0D 20
0070   06 3E CD A5 07 D5 CD 49 09 CD BF 02 CD 68 02 7C
0080   B5 C1 CA 16 09 1B 7C 12 1B 7D 12 C5 D5 79 93 F5
0090   CD ED 07 D5 C2 A7 00 D5 CD 0B 08 C1 2A 22 20 CD
00A0   93 08 60 69 22 22 20 C1 2A 22 20 F1 E5 FE 03 CA
00B0   4D 00 85 6F 3E 00 8C 67 CD 51 09 CD 62 02 D2 9E
00C0   07 22 22 20 D1 CD 9E 08 D1 E1 CD 93 08 C3 70 00
00D0   4D 49 4B 52 4F 20 42 41 53 49 43 00 4C 49 53 54
00E0   00 53 03 4C 4C 49 53 54 00 50 03 52 55 4E 00 10
00F0   03 4E 45 57 00 07 03 4D 4F 4E 49 54 4F 52 00 9C
```

```
HEXADECIMALNI VYPIS EPROM JPR-1                    STRANA  2

0100    0E 52 41 4D 00 71 09 4C 4F 41 44 00 1F 0B 53 41
0110    56 45 00 95 0B 4E 45 58 54 00 A5 04 4C 45 54 00
0120    97 05 43 4C 53 00 10 09 43 4C 45 41 52 00 0C 05
0130    48 41 52 44 00 44 03 44 49 53 50 4C 00 4A 03 49
0140    46 00 1C 05 47 4F 54 4F 00 33 03 47 4F 53 55 42
0150    00 03 04 52 45 54 55 52 4E 00 25 04 52 45 4D 00
0160    06 05 46 4F 52 00 40 04 49 4E 50 55 54 00 37 05
0170    50 52 49 4E 54 00 A1 03 4C 50 52 49 4E 54 00 9E
0180    03 45 4E 44 00 13 00 53 54 4F 50 00 0A 0B 43 41
0190    4C 4C 00 9F 09 4F 55 54 43 48 41 52 00 C3 0A 4F
01A0    55 54 00 AC 09 4F 24 00 33 0A 49 24 00 40 0A 57
01B0    41 49 54 00 D2 09 42 45 45 50 00 E3 09 4F 4E 00
01C0    00 23 44 49 00 80 23 50 4F 4B 45 00 79 0A 4D 41
01D0    53 4B 00 67 09 54 41 42 00 FA 09 42 59 54 45 00
01E0    8E 0A 57 4F 52 44 00 98 0A 00 91 05 52 4E 44 00
01F0    B5 06 41 42 53 00 E2 06 53 49 5A 45 00 EB 06 50

HEXADECIMALNI VYPIS EPROM JPR-1                    STRANA  3

0200    45 45 4B 00 72 0A 49 4E 43 48 41 52 00 CD 0A 48
0210    45 58 00 D4 0A 49 4E 4D 00 10 0A 49 4E 00 0B 0A
0220    27 00 B5 0A 54 4F 50 00 03 00 4C 45 4E 00 0B 00
0230    00 93 06 54 4F 00 50 04 00 69 07 53 54 45 50 00
0240    5C 04 00 62 04 3E 3D 00 AF 05 23 00 B5 05 3E 00
0250    BB 05 3D 00 CA 05 3C 3D 00 C2 05 3C 00 D0 05 00
0260    D6 05 7C BA C0 7D BB C9 1A FE 20 C0 13 C3 68 02
0270    F1 CD 50 07 C3 69 07 CD 68 02 D6 40 D8 C2 9B 02
0280    13 CD A4 06 29 DA E9 02 D5 EB CD EB 06 CD 62 02
0290    DA 9F 07 2A E7 20 CD 0E 07 D1 C9 FE 1B 3F D8 13
02A0    21 B1 20 07 85 6F 3E 00 8C 67 C9 E3 CD 68 02 BE
02B0    23 CA BB 02 C5 4E 06 00 09 C1 1B 13 23 E3 C9 21
02C0    00 00 44 CD 68 02 FE 30 D8 FE 3A D0 3E F0 A4 C2
02D0    E9 02 04 C5 44 4D 29 29 09 29 1A 13 E6 0F 85 6F
02E0    3E 00 8C 67 C1 1A F2 C6 02 D5 11 F0 02 C3 6D 07
02F0    48 4F 57 3F 0D 52 45 41 44 59 0D 57 48 41 54 3F

HEXADECIMALNI VYPIS EPROM JPR-1                    STRANA  4

0300    0D 53 4F 52 52 59 0D CD 63 07 21 ED 20 22 22 20
0310    CD 63 07 11 ED 20 21 00 00 CD F5 07 DA 4D 00 EB
0320    22 0B 20 EB 13 13 AF 32 21 20 CD 02 0B 21 14 01
0330    C3 19 09 CD A5 05 D5 CD 63 07 CD ED 07 C2 EA 02
0340    F1 C3 1F 03 CD F3 0B CD 70 02 CD FA 0B CD 70 02
0350    CD F3 0B CD BF 02 3E FF 32 06 20 CD 68 02 FE 0D
0360    C2 81 03 CD ED 07 DA 98 03 3A 06 20 FE FF CA 75
0370    03 3D CA 4D 00 32 06 20 CD 7E 0B CD F5 07 C3 66
0380    03 FE 2B C2 63 03 13 CD 68 02 E5 CD BF 02 7D 3C
0390    3C 32 06 20 E1 C3 63 03 CD FA 0B C3 50 00 CD F3
03A0    0B 0E 08 CD AB 02 3A 09 CD F2 08 CD FA 0B C3 26
03B0    03 CD AB 02 0D 09 CD F2 08 CD FA 0B C3 16 03 CD
03C0    AB 02 2A 0E CD A5 05 7D 0F 0F E6 C0 32 21 20 C3
03D0    E4 03 CD AB 02 23 07 CD A5 05 4D C3 E4 03 CD 25
03E0    08 C3 F8 03 CD AB 02 2C 06 CD 50 07 C3 BF 03 CD
03F0    F2 08 CD FA 0B CD 70 02 CD A5 05 C5 CD 3A 08 C1

HEXADECIMALNI VYPIS EPROM JPR-1                    STRANA  5

0400    C3 E4 03 CD C9 08 CD A5 05 D5 CD ED 07 C2 EA 02
0410    2A 0B 20 E5 2A 0D 20 E5 21 00 00 22 13 20 39 22
0420    0D 20 C3 1F 03 CD 63 07 2A 0D 20 7C B5 CA 69 07
0430    F9 E1 22 0D 20 E1 22 0B 20 D1 CD AD 08 CD 70 02
0440    CD C9 08 CD 37 07 2B 22 13 20 21 32 02 C3 19 09
0450    CD A5 05 22 17 20 21 3A 02 C3 19 09 CD A5 05 C3
0460    65 04 21 01 00 22 15 20 2A 0B 20 22 19 20 EB 22
0470    1B 20 01 0A 00 2A 13 20 EB 60 68 39 3E 09 7E 23
0480    B6 CA 9E 04 7E 2B BA C2 7D 04 7E BB C2 7D 04 EB
0490    21 00 00 39 44 4D 21 0A 00 19 CD 9E 08 F9 2A 1B
04A0    20 EB CD 70 02 CD 77 02 DA 69 07 22 0F 20 D5 EB
04B0    2A 13 20 7C B5 CA 6A 07 CD 62 02 CA C8 04 D1 CD
04C0    AD 08 2A 0F 20 C3 AE 04 5E 23 56 2A 15 20 E5 7C
04D0    AA 7A 19 FA DA 04 AC FA FE 04 EB 2A 13 20 73 23
04E0    72 2A 17 20 F1 B7 F2 EA 04 EB CD 2D 07 D1 DA 00
04F0    05 2A 19 20 22 0B 20 2A 1B 20 EB CD 70 02 E1 D1
```

Jak pracuje interpreter?

Projde-li program po zapnutí počítače úseky studeného a horkého startu, je vypsán vodicí znak a interpreter čeká na vaše příkazy nebo povely. Text příkazů nebo povelů je ukládán do vstupního řádkového bufferu, kde je ho možno upravovat až do stisku klávesy CR. Jak již bylo řečeno, je tento buffer umístěn vždy na konci celé paměti RAM vyhrazené interpreteru a má délku 64 bytů. Proto také můžete psát programové řádky až do délky 64 znaků. Přitom pozor, protože klávesou BS se z druhého řádku na první již nedostanete a musíte napsat programový řádek celý znovu. Podprogram pro vstup textu do řádkového bufferu začíná na adrese 7A5H a končí na adrese 7ECH a lze jej použít i mimo rámec interpreteru. Pokud nerozšíříme paměť pomocí příkazu RAM, najdeme začátek vstupního bufferu na adrese 24ADH.

Ukončíme-li vstup příkazu nebo povelu klávesou CR, vrátí se program na adresu 75H do základní programové smyčky interpreteru, která leží mezi adresami 70H a CFH. V této smyčce se nejprve rozhodne, jedná-li se o programový řádek začínající číslem, nebo o příkazový řádek v přímém módu, či povel.

V druhém případě pokračuje interpreter na adrese 916H přímo prohledáním tabulky klíčových slov a skokem do příslušné rutiny. Jde-li o programový řádek, pokračuje program na adrese 85H a řádek je zařazen, vymazán nebo přepsán v programovém bufferu, začínajícím na adrese 20EDH. Přitom je každý nový řádek pečlivě zatříděn podle pořadového čísla řádku.

Podíváte-li se v monitoru do programového bufferu pro napsání krátkého programu v BASIC, zjistíte, že každý řádek začíná dvěma byty s binární hodnotou čísla řádku, následuje nezkrácený text a celý řádek končí znakem CR (hexadecimálně 0D).

Konec programu v programovém bufferu není nijak označen, zjistíte ho podle údaje na adrese 2022H, udávající první volnou adresu bufferu. Šikovný programátor může těchto informací využít například k přepsání čísla programového řádku pomocí monitoru, kde lze také částečně program i editovat, nezměníme-li počet písmen. Přepsáním binární hodnoty čísla řádku však nesmíme porušit sled programových řádků, které interpreter setřiďuje při jejich ukládání.

Přepsáním údaje na adrese 2022H (dva byty) před použitím příkazu SAVE můžeme interpreter „podvést" a nechat ho vyděrovat i úsek paměti RAM za koncem programu v BASIC. To může být užitečné například pro úschovu programů ve strojovém kódu. Nesmíme však zapomenout při opětovném načtení programu příkazem LOAD vrátit původní hodnotu na adresy 2022H a 2023H. Jinak by interpreter považoval celý úsek paměti za programový buffer obsazený textem programu a výpis pomocí povelu LIST by nás asi neuspokojil.

Vraťme se však k místu, kdy interpreter opouští základní programovou smyčku po napsání povelu a vyberme si rovnou povel RUN.

Realizace programu v BASIC

Realizace programu po povelu RUN má velmi jednoduchá pravidla: v programovém bufferu je postupně prohlížen text programu, přičemž jako ukazatel slouží dvojitý registr D. Ten obsahuje vždy adresu právě zpracovávaného znaku v textu programu, zatímco číselné údaje a parametry jsou mezi příkazy a funkcemi předávány pomocí dvojitého registru H.

Prohlížení textu začíná nalezením čísla prvního programového řádku. Adresa v registru D je dvakrát inkrementována a protože interpreter předpokládá nalezení některého klíčového slova příkazu, je prohledávána tabulka klíčových slov počínaje příkazem NEXT. Část tabulky, obsahující povely, je tedy záměrně přeskočena. Všechny mezery v textu jsou vynechány, nesmí se však vyskytnout uprostřed klíčových slov.

Je-li v tabulce nalezeno odpovídající klíčové slovo, končící bytem s hodnotou 0, pokračuje interpreter provedením příslušné rutiny v příkazu, jejíž adresa je v tabulce v následujících dvou bytech. Pro příkaz NEXT je například klíčové slovo od adresy 115H, nulový byte je na adrese 119H a v dalších dvou bytech je adresa výkonné rutiny příkazu, tedy 4A5H.

Na konci každého příkazu interpreter zjistí, končí-li příkaz dvojtečkou nebo znakem CR a rozhodne, bude-li pokračovat hledáním dalšího klíčového slova příkazu, nebo hledáním dalšího čísla programového řádku. V obou případech je ještě testováno stisknutí tlačítka INT, které umožňuje přerušit realizaci programu.

Realizace jednotlivých příkazů

Označíte-li si v hexadecimálním výpisu programu začátky a konce rutin jednotlivých příkazů (po dekódování tabulky klíčových slov), překvapí vás jistě jejich malá délka. Příkaz LET má například 14 bytů. Je to proto, že celý interpreter je navržen velice efektivně a využívá několika standardních podprogramů. Jedním z nejdůležitějších je například podprogram pro vyhodnocení našeho „cokoli". Podprogram začíná na adrese 5A5H a končí na adrese 6A3H.

Vždy, když některý příkaz očekává zadání číselné hodnoty, stačí „zavolat" tento podprogram a ve dvojitém registru H se vrátí výsledek, ať již je zadání ve tvaru čísla, funkce nebo např. textového operátoru. Při jakékoli chybě opustí interpreter předčasně tento podprogram a vypíše chybný programový řádek s otazníkem před znakem, který porušil některé pravidlo o syntaxi příkazů. Velmi často to bývá nesprávný počet závorek, chyba v klíčovém slově nebo chybějící parametr.

Pokusí-li se někdo i přes mé varování přeložit strojový kód pomocí reverzního assembleru, zcela jistě narazí na řadu nejasností, způsobených právě podprogramem pro kontrolu syntaxe programových příkazů. Tento podprogram je umístěn od adresy 2ABH a končí na adrese 2BEH. Při jeho volání je použit malý programátorský trik. Podívejme se například na podprogram, kontrolující umístění závorek u parametru funkcí.

Podprogram leží na adrese 6A4H a začíná voláním jmenovaného podprogramu pro kontrolu syntaxe. Toto volání však má svoje zvláštní pravidla. V dalším bytu za instrukcí CALL musí totiž ležet znak kódu ASCII, který podle pravidel syntaxe očekáváme a v následujícím bytu hodnota, udávající počet bytů, které program přeskočí, není-li tento znak nalezen. V našem konkrétním příkladu má byte za instrukcí volání hodnotu 28H, což odpovídá levé závorce, a další byte hodnotu 9. (A skutečně o 9 byte dále leží skok na chybové hlášení WHAT. To v případě, nebude-li správná závorka nalezena.)

Po vyhodnocení výrazu podprogramem na adrese 5A5H je stejný trik použit pro kontrolu pravé závorky (29H) a skok má délku 1 byte.

Programátor, který by se dal svést reverzním překladem tohoto úseku programu, by narazil na nejasnost ihned při dekódování čtvrtého byte s hodnotou

```
HEXADECIMALNI VYPIS EPROM JPR-1                    STRANA  6

0500    CD AD 08 CD 70 02 21 00 00 C3 1F 05 21 B1 20 D5
0510    11 36 00 0E 00 CD 8A 0C D1 CD 70 02 CD A5 05 7C
0520    B5 C2 26 03 CD 0D 08 D2 1F 03 C3 4D 00 2A 11 20
0530    F9 E1 22 0B 20 D1 D1 D5 CD 25 08 C3 47 05 CD 77
0540    02 DA 85 05 C3 59 05 D5 CD 77 02 DA 69 07 1A 4F
0550    97 12 D1 CD 17 08 79 1B 12 D5 EB 2A 0B 20 E5 21
0560    37 05 22 0B 20 21 00 00 39 22 11 20 D5 06 3A CD
0570    A5 07 CD 49 09 CD A5 05 D1 EB 73 23 72 E1 22 0B
0580    20 CD 02 0B D1 F1 CD AB 02 2C 03 C3 37 05 CD 70
0590    02 1A FE 0D CA A2 05 CD 37 07 CD AB 02 2C 03 C3
05A0    97 05 CD 70 02 CD ED 05 E5 21 44 02 C3 19 09 CD
05B0    D8 05 D8 6F C9 CD D8 05 C8 6F C9 CD D8 05 C8 D8
05C0    6F C9 CD D8 05 6F C8 D8 6C C9 CD D8 05 C0 6F C9
05D0    CD D8 05 D0 6F C9 E1 C9 79 E1 C1 E5 C5 4F CD ED
05E0    05 EB E3 CD 2D 07 D1 21 00 00 3E 01 C9 CD AB 02
05F0    2D 06 21 00 00 C3 1F 06 CD AB 02 2B 00 CD 29 06

HEXADECIMALNI VYPIS EPROM JPR-1                    STRANA  7

0600    CD AB 02 2B 15 E5 CD 29 06 EB E3 7C AA 7A 19 D1
0610    FA 00 06 AC F2 00 06 C3 E9 02 CD AB 02 2D 92 E5
0620    CD 29 06 CD 18 07 C3 09 06 CD 8D 06 CD AB 02 2A
0630    2D E5 CD 8D 06 06 00 CD 15 07 E3 CD 15 07 EB E3
0640    7C B7 CA 4B 06 7A B2 EB C2 EA 02 7D 21 00 00 B7
0650    CA 7F 06 19 DA EA 02 3D C2 53 06 C3 7F 06 CD AB
0660    02 2F 4E E5 CD 8D 06 06 00 CD 15 07 E3 CD 15 07
0670    EB E3 EB 7A B3 CA EA 02 C5 CD F8 06 60 69 C1 D1
0680    7C B7 FA E9 02 78 B7 FC 18 07 C3 2C 06 21 EB 01
0690    C3 19 09 CD 77 02 DA 9E 06 7E 23 66 6F C9 CD BF
06A0    02 78 B7 C0 CD AB 02 28 09 CD A5 05 CD AB 02 29
06B0    01 C9 C3 69 07 CD A4 06 7C B7 FA E9 02 B5 CA E9
06C0    02 D5 E5 2A 1D 20 11 9D 0F CD 62 02 DA D2 06 21
06D0    62 02 5E 23 56 22 1D 20 E1 EB C5 CD F8 06 C1 D1
06E0    23 C9 CD A4 06 1B CD 15 07 13 C9 2A 22 20 D5 EB
06F0    2A E7 20 CD 0E 07 D1 C9 E5 6C 26 00 CD 03 07 41

HEXADECIMALNI VYPIS EPROM JPR-1                    STRANA  8

0700    7D E1 67 0E FF 0C CD 0E 07 D2 05 07 19 C9 7D 93
0710    6F 7C 9A 67 C9 7C B7 F0 7C B5 C8 7C F5 2F 67 7D
0720    2F 6F 23 F1 AC F2 E9 02 78 EE 80 47 C9 7C AA F2
0730    33 07 EB CD 62 02 C9 CD 77 02 DA 69 07 E5 CD AB
0740    02 3D 0A CD A5 05 44 4D E1 71 23 70 C9 C3 69 07
0750    CD AB 02 3A 04 F1 C3 26 03 CD AB 02 0D 04 F1 C3
0760    16 03 C9 CD 68 02 FE 0D C8 D5 11 FB 02 97 32 21
0770    20 CD 17 08 CD FA 0B D1 1A F5 97 12 2A 0B 20 E5
0780    7E 23 B6 D1 CA 4D 00 7E B7 FA 2D 05 CD 7E 08 1B
0790    F1 12 3E 3F CD F4 08 97 CD 17 08 C3 4D 00 D5 11
07A0    01 03 C3 6D 07 78 CD F4 08 CD 49 09 CD F8 0C FE
07B0    08 CA C8 07 FE 7F CA D5 07 CD F4 08 12 13 FE 0D
07C0    C8 7B CD 59 09 C2 AC 07 7B CD 60 09 CA D5 07 CD
07D0    DD 07 C3 AC 07 CD F2 08 06 3E C3 A5 07 1B 3E 08
07E0    CD F4 08 3E 20 CD F4 08 3E 08 C3 F4 08 7C B7 FA
07F0    E9 02 11 ED 20 E5 2A 22 20 2B CD 62 02 E1 D8 1A

HEXADECIMALNI VYPIS EPROM JPR-1                    STRANA  9

0800    95 47 13 1A 9C DA 0C 08 1B B0 C9 13 13 1A FE 0D
0810    C2 0C 08 13 C3 F5 07 47 1A 13 B8 C8 CD F4 08 FE
0820    0D C2 18 08 C9 CD AB 02 22 0F 3E 22 CD 17 08 FE
0830    0D E1 CA B9 03 23 23 23 E9 C9 06 00 CD 15 07 F2
0840    45 08 06 2D 0D D5 11 0A 00 D5 0D C5 CD F8 06 78
0850    B1 CA 5C 08 E3 2D E5 60 69 C3 4C 08 C1 0D 79 B7
0860    FA 6B 08 3E 20 CD F4 08 C3 5D 08 78 B7 C4 F4 08
0870    5D 7B FE 0A D1 C8 C6 30 CD F4 08 C3 71 08 1A 6F
0880    13 1A 67 13 0E 04 CD 3A 08 3E 20 CD F4 08 97 CD
0890    17 08 C9 CD 62 02 C8 1A 02 13 03 C3 93 08 78 92
08A0    C2 A6 08 79 93 C8 1B 2B 1A 77 C3 9E 08 C1 E1 22
08B0    13 20 7C B5 CA C7 08 E1 22 15 20 E1 22 17 20 E1
08C0    22 19 20 E1 22 1B 20 C5 C9 21 4C 20 CD 18 07 C1
08D0    39 D2 9E 07 2A 13 20 7C B5 CA EF 08 2A 1B 20 E5
08E0    2A 19 20 E5 2A 17 20 E5 2A 15 20 E5 2A 13 20 E5
08F0    C5 C9 3E 0D F5 FE 0D CA FF 08 CD 07 20 F1 C9 CD
```

```
HEXADECIMALNI VYPIS EPROM JPR-1                    STRANA  10

0900    05 09 C3 FD 08 3E 0D CD 07 20 3E 0A CD 07 20 C9
0910    CD 75 0C CD 70 02 21 DB 00 CD 68 02 D5 1A 13 FE
0920    2E CA 3B 09 23 BE CA 1D 09 3E 00 1B BE CA 42 09
0930    23 BE C2 30 09 23 23 D1 C3 19 09 3E 00 23 BE C2
0940    3D 09 23 7E 23 66 6F F1 E9 E5 2A E9 20 54 5D E1
0950    C9 E5 2A E7 20 54 5D E1 C9 E5 2A EB 20 BD E1 C9
0960    E5 2A E9 20 BD E1 C9 CD A5 05 7D 32 0A 20 CD 70
0970    02 CD A5 05 EB 21 AB 23 EB CD 62 02 DA 9F 07 7C
0980    B7 FA 9F 07 7E 2F 77 46 B8 C2 9F 07 22 EB 20 7D
0990    D6 40 6F 7C DE 00 67 22 E9 20 2B 2B C3 4A 00 CD
09A0    A5 05 D5 01 AB 09 C5 E9 D1 CD 70 02 CD A5 05 E5
09B0    CD AB 02 2C 1A CD A5 05 45 3E D3 32 03 20 E1 7D
09C0    32 04 20 3E C9 32 05 20 78 CD 03 20 CD 70 02 C3
09D0    69 07 CD A5 05 D5 26 64 CD EF 09 2D C2 D6 09 D1
09E0    CD 70 02 CD A5 05 D5 EB CD F8 0D D1 CD 70 02 16
09F0    64 15 C2 F1 09 25 C8 C3 EF 09 CD A4 06 7C B5 CC

HEXADECIMALNI VYPIS EPROM JPR-1                    STRANA  11

0A00    70 02 2B 3E 20 CD 07 20 C3 FD 09 CD 1A 0A 6F C9
0A10    CD 1A 0A 6F 3A 0A 20 A5 6F C9 CD A4 06 E5 3E DB
0A20    32 03 20 E1 7D 32 04 20 3E C9 32 05 20 CD 03 20
0A30    26 00 C9 CD A5 05 D5 EB AF CD 17 0B D1 CD 70 02
0A40    CD A5 05 D5 EB 2A 22 20 EB CD 62 02 DA 9F 07 CD
0A50    49 09 CD AC 07 44 4D EB 2B CD 49 09 D5 CD 93 0B
0A60    AF 02 D1 23 CD 0E 07 EB 21 01 20 73 23 72 D1 CD
0A70    70 02 CD A4 06 6E 26 00 C9 CD A5 05 E5 CD AB 02
0A80    2C 09 CD A5 05 7D E1 77 CD 70 02 C3 69 07 CD A4
0A90    06 7D CD A1 0A CD 70 02 CD A4 06 CD 02 0E CD 70
0AA0    02 F5 0F 0F 0F 0F CD AA 0A F1 E6 0F C6 90 27 CE
0AB0    40 27 C3 F4 0B 1A 13 6F 26 00 CD AB 02 27 01 C9
0AC0    C3 69 07 CD A5 05 7D CD 07 20 CD 70 02 CD F8 0C
0AD0    26 00 6F C9 C5 21 00 00 CD AB 02 28 1D 1A FE 0D
0AE0    CA 69 07 CD 23 0E DA 69 07 29 29 29 29 06 00 4F
0AF0    09 13 CD AB 02 29 03 C3 00 0B C3 DD 0A C3 69 07

HEXADECIMALNI VYPIS EPROM JPR-1                    STRANA  12

0B00    C1 C9 3A 00 28 07 D8 C3 0D 0B CD 63 07 2A 0B 20
0B10    EB CD F2 08 3E 2A CD F4 08 CD 7E 08 C3 4D 00 21
0B20    ED 20 CD 7F 0B FE 00 CA 22 0B F5 11 DA 0B 97 CD
0B30    17 08 F1 CD 07 20 CD 7F 0B FE 0D C2 33 0B CD 7F
0B40    0B 5F CD 7F 0B 57 21 ED 20 CD 62 02 CA 57 0B CD
0B50    7F 0B 77 23 CA 49 0B 22 22 20 C3 4D 00 06 96 AF
0B60    CD 68 0B 05 C2 5F 0B C9 2F D3 43 3E 01 D3 42 3E
0B70    30 3D C2 71 0B AF D3 42 DB 41 07 D2 78 0B C9 DB
0B80    40 DB 41 E6 01 C2 81 0B DB 40 2F F5 DB 41 E6 01
0B90    CA 8C 0B F1 C9 CD BB 0B 2A 22 20 7D CD 68 0B 7C
0BA0    CD 68 0B EB 21 ED 20 CD 62 02 CA B5 0B 7E CD 68
0BB0    0B 23 C3 A7 0B CD 5D 0B C3 4D 00 11 DA 0B 97 CD
0BC0    17 08 06 3D CD A5 07 CD 5D 0B CD 49 09 1A F5 CD
0BD0    68 0B F1 FE 0D C8 13 C3 CD 0B 4E 41 4D 45 20 00
0BE0    CD 06 0C E6 7F F5 32 00 2C 3A 00 2C E6 80 C2 E9
0BF0    0B F1 C9 21 E0 0B 22 08 20 C9 3E C3 32 07 20 21

HEXADECIMALNI VYPIS EPROM JPR-1                    STRANA  13

0C00    06 0C 22 08 20 C9 F5 E5 D5 C5 4F CD D9 0C 3A 21
0C10    20 47 7E E6 C0 FE 80 C2 23 0C 78 B7 C2 23 0C 7E
0C20    E6 7F 77 79 FE 0D CA A0 0C FE 08 CA 4B 0C FE 0A
0C30    CA A7 0C E6 3F 5F 78 B3 77 E6 C0 FE C0 CC 5A 0C
0C40    CD 5A 0C CD 93 0C C1 D1 E1 F1 C9 3A 20 20 3D FE
0C50    FF CA 43 0C 32 20 20 C3 43 0C 3A 20 20 3C 32 20
0C60    20 FE 28 D8 AF 32 20 20 3A 1F 20 3C FE 18 CA AD
0C70    0C 32 1F 20 C9 F5 E5 D5 C5 21 00 38 11 00 08 CD
0C80    B8 0C CD F0 0C C3 43 0C 0E 20 71 23 1B 7A B3 C2
0C90    8A 0C C9 3A 21 20 B7 C0 CD D9 0C 7E F6 80 77 C9
0CA0    AF 32 20 20 C3 43 0C CD 68 0C C3 43 0C 21 00 38
0CB0    11 40 38 0E 17 CD CE 0C C5 01 18 00 09 EB 09 EB
0CC0    C1 0D C2 B5 0C 21 C0 3D 11 40 00 C3 88 0C 06 28
0CD0    1A 77 13 23 05 C2 D0 0C C9 26 00 3A 1F 20 6F 06
0CE0    06 29 05 C2 E1 0C 3A 20 20 B5 6F 7C F6 38 67 C9
0CF0    AF 32 1F 20 32 20 20 C9 C5 D5 E5 0E 80 16 05 06
```

28H. Operační kód pro tuto hodnotu u CPU 8080 totiž neexistuje.

Zpracování funkcí

Narazí-li podprogram pro vyhodnocení výrazů na klíčové slovo funkce, zabezpečí prohledání tabulky klíčových slov od adresy 1ECH. Další postup je stejný jako při zpracování příkazů. Podprogram pro zpracování funkce převezme příslušný parametr, zpracuje ho a vrátí výslednou hodnotu v dvojitém registru H. Rozdíl je pouze v tom, že podprogramy příkazů končí přechodem na hledání dalšího příkazu nebo programového řádku, zatímco podprogramy funkcí končí instrukcí pro návrat z podprogramu RET. To proto, že zpracování funkcí je vždy součástí vyhodnocení výrazu. Z toho také plyne to, že místo výrazu může být naše „cokoli".

Úpravy interpreteru

Je jasné, že se interpreter Mikro BASIC neubrání vašim zásahům. Je to také dobře, protože každá úprava předpokládá důkladnou znalost celého interpreteru a jeho pochopení je ideálním cvičením pro středně a více pokročilé programátory.

Úpravou interpreteru je možné přizpůsobit repertoár příkazů a funkcí a vyrobit si také svůj vlastní jazyk, problémově orientovaný do určité aplikační oblasti. Postup je jednoduchý.

Do tabulky klíčových slov zařadíme nové klíčové slovo v kódu ASCII, za něj umístíme nulový byte a dále adresu podprogramu, který příslušný povel, příkaz nebo funkci realizuje. Přitom pozor, nový povel musíme v tabulce zařadit mezi povely, příkaz mezi příkazy a stejně tak novou funkci. Zpracování výrazů a předávání parametrů snadno „odkoukáme" od existujících podprogramů. U příkazů nesmíme hlavně zapomenout uschovat registr D, který slouží jako ukazatel do programového textu.

Přeadresování interpreteru

Ne každému bude vyhovovat překlad interpreteru od adresy nula, určený pro paměti typu 2716 s využitím paměti RAM 1K od adresy 2000H. Jak již bylo řečeno dříve, vyžaduje použití pamětí typu 2708, pokud jsou na desce JPR-1, nový překlad programu. Opět nedoporučuji amatérskou úpravu ve strojovém kódu, která by si vyžádala neúměrnou námahu.

Program interpreteru musí být totiž rozdělen na čtyři úseky, začínající na adresách s roztečí 2K o délce úseku 1K. To však vyžaduje mít k dispozici zdrojový text programu, upravit jej pomocí editoru a znovu přeložit pomocí překladače. Vložené mezery o délce 1K však musí být umístěny tak, aby nenarušily činnost programu, a bude lepší, necháte-li to na nás. Úpravami a rozšiřováním programového vybavení mikropočítače JPR-1 se budeme zabývat jak v Městské stanici mladých techniků v Praze, tak v řadě ZO Svazarmu, konkrétně např. v ZO 602 v Praze 6, a redakce AR nám i vám bude jistě vycházet vstříc. Záměrně píši „vám", protože předpokládám, že se brzy objeví i vaše programy.

Daleko snazší je použít desku pamětí REM-1, u níž je možno nastavit rozteč adresace 1K a lze tedy použít zde uvedený překlad programu pro paměti typu 2708. V tomto případě však bude nutné přemístit paměť RAM také na desku REM-1.

Jestliže se však neuspokojíte s maximální délkou programového bufferu 702 byte, bude nutné použít větší paměti od adresy 4000H a celý interpreter přeadresovat v místech, kde pracuje s pamětí RAM.

Předpokládám, že tuto práci může zvládnout zručný programátor i ručně. Přeadresování se týká všech těch instrukcí, které mají v druhém a třetím byte adresu v rozsahu 2000H až 23EDH. Mimo instrukcí LXI, LHLD, SHLD, STA, LDA se to týká také instrukcí CALL 2007H, JMP 23EEH a JMP 23F7H.

Poslední dvě instrukce se nacházejí na adresách 8 a 10H a umožňují používat instrukce RST 1 a RST 2, případně odpovídající přerušení i přesto, že interpreter je umístěn od začátku paměťového prostoru. Příslušný skok při přerušení je takto přenesen do paměti RAM, kam můžeme vložit vhodný skok do uživatelského programu.

Několik rad závěrem

Všechny rady by měly být dávány podle toho, komu jsou určeny. Jiné potřebuje úplný začátečník v programování, jiné ten, kdo s mikropočítači pracuje, a jiné ten, kdo je programátorem u velkého počítače. Ty následující jsou určeny těm méně zkušeným.

Předně se snažte dostat se co nejdříve k nějakému mikropočítači. Programovat se totiž nejlépe naučíte přímo u počítače. Jedna hodina hraní s počítačem vydá za deset hodin studia literatury. Snažte se nejdříve osvojit si ty nejjednodušší instrukce a příkazy a zkoušejte si jejich použití v nejrůznějších variantách. Studujte publikované programy a snažte se jim rozumět. Když narazíte na nějakou nejasnost, vraťte se k teorii a výsledek si ověřte opět u počítače. Jinak to opět zapomenete.

Nebojte se toho, že existuje řada variant jazyka BASIC. Kolik jazyků umíš, tolikrát si programátorem. Porovnávejte jednotlivé varianty a snažte se vhodně nahradit ty příkazy, které např. Mikro BASIC nemá. Sbírejte si do zásoby nejrůznější programátorské triky a snažte se jim přijít na kloub.

Před tím, než začnete psát větší program, si důkladně promyslete jeho strukturu. Snažte se rozdělit si problematiku do relativně samostatných částí. Tyto části řešte pomocí podprogramů a vhodným způsobem zvolte předávání parametrů a výsledků. Správnou funkci každého podprogramu si ověřte dříve, než jej zapojíte do celého programu. Všechny parádičky a vylepšování programu si nechte na konec.

Pište si co nejvíce poznámek, i když třeba jen do sešitu. Programy si uschovejte a úspěšné úseky programu používejte v dalších programech.

Jestliže se příliš „zamotáte" při psaní programu, raději se vraťte a začněte znovu a jinak. Nevzdávejte se. A nikdy nedávejte vinu počítači. Téměř vždy se nakonec ukáže, že chyba byla na vaší straně.

HEXADECIMALNI VYPIS EPROM JPR-1 STRANA 14

```
0D00  1E 26 00 3A 00 20 E6 E0 B0 32 00 24 78 37 17 E6
0D10  1F 47 3A 00 24 FE FF C4 4E 0D 15 C2 03 0D CD 74
0D20  0D 79 FE 01 CA 32 0D DA FD 0C FE 80 CA 43 0D C3
0D30  FB 0C 7D 84 21 89 0D 4F 06 00 09 CD F5 0D 7E E1
0D40  D1 C1 C9 0E 90 0D C2 45 0D 0E 00 C3 FD 0C 1E 08
0D50  0F D4 59 0D 1D C2 50 0D C9 F5 7A 3D 07 07 07 E6
0D60  38 1D B3 1C FE 07 CA 6D 0D 6F 0C F1 C9 26 28 CD
0D70  80 0D F1 C9 3A 00 20 E6 D0 32 00 20 32 00 24 C9
0D80  3A 00 20 F6 20 C3 79 0D C9 30 50 0D 08 31 51 41
0D90  0E 39 4F 4C 20 32 57 53 5A 38 49 4B 4D 33 45 44
0DA0  58 37 55 4A 4E 34 52 46 43 36 59 48 42 35 54 47
0DB0  56 29 7F 0D 08 21 51 41 0E 28 23 3D 20 3F 57 53
0DC0  5A 3E 40 2A 2E 22 45 44 58 3C 3A 2D 2C 24 52 46
0DD0  43 27 2F 2B 3B 5E 54 47 56 C5 3A 00 20 F6 40 32
0DE0  00 24 0E 42 0D C2 E4 0D E6 B0 32 00 24 0E 50 0D
0DF0  C2 EF 0D C1 C9 11 40 00 CD D9 0D 1B 7A B3 C2 F8
```

HEXADECIMALNI VYPIS EPROM JPR-1 STRANA 15

```
0E00  0D C9 7C CD A1 0A 7D C3 A1 0A 21 00 00 CD F8 0C
0E10  4F CD 07 20 CD 23 0E DA 33 0E 29 29 29 29 B5 6F
0E20  C3 0D 0E D6 30 D8 FE 17 3F D8 FE 0A 3F D0 D6 07
0E30  FE 0A C9 79 FE 0D CA 44 0E FE 20 C8 3E 3F CD 07
0E40  20 C3 9F 0E F5 3E 0A CD 07 20 F1 4F C9 06 00 E5
0E50  D1 7E FE C3 CA 94 0E FE CD CA 94 0E E6 EF FE 22
0E60  CA 94 0E FE 2A CA 94 0E E6 CF FE 01 CA 94 0E E6
0E70  C7 FE C2 CA 94 0E FE C4 CA 94 0E 7E E6 F7 FE D3
0E80  CA 95 0E E6 E7 E6 C7 FE 06 CA 95 0E FE C6 CA 95
0E90  0E C3 96 0E 04 04 04 EB C9 CD 75 0C 31 AF 20 CD
0EA0  05 09 11 F7 00 97 CD 17 08 CD 05 09 3E 2A CD 07
0EB0  20 CD F8 0C CD 07 20 FE 44 CA E0 0E FE 53 CA D8
0EC0  0E FE 43 CA 99 0E FE 47 CA DC 0E FE 42 CA 19 00
0ED0  FE 52 CA 4D 00 C3 3C 0E 21 9F 0E E5 CD 0A 0E E9
0EE0  CD 0A 0E CD 05 09 CD 02 0E CD 4D 0E C5 E5 3E 20
0EF0  CD 07 20 7E CD A1 0A 23 05 C2 EE 0E D1 C1 3E 04
```

HEXADECIMALNI VYPIS EPROM JPR-1 STRANA 16

```
0F00  90 47 07 E6 0F 80 47 3E 20 CD 07 20 05 C2 07 0F
0F10  3E 3A CD 07 20 1A E6 7F FE 20 DA 25 0F FE 7F CA
0F20  25 0F C3 27 0F 3E 20 CD 07 20 3E 3A CD 07 20 CD
0F30  F8 0C 4F CD 07 20 FE 08 CA 6A 0F FE 20 CA 6F 0F
0F40  FE 0D CA E3 0E FE 2E CA 9F 0E FE 22 CA 77 0F E5
0F50  21 00 00 CD 14 0E 47 7D 12 13 78 E1 FE 20 CA 2F
0F60  0F FE 0D C2 3C 0E EB C3 E6 0E 1B EB C3 E3 0E EB
0F70  23 C3 E3 0E CD DD 07 CD F8 0C FE 0D CA 97 0F CD
0F80  07 20 FE 22 CA 9D 0F FE 08 CA 74 0F 4F 12 1A B9
0F90  C2 3C 0E 13 C3 77 0F CD 05 09 C3 77 0F CD F8 0C
0FA0  FE 20 CA 2F 0F EB C3 E3 0E CA D9 FF 00 FF 00 FF
0FB0  00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
0FC0  00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
0FD0  00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
0FE0  00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
0FF0  00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF
```

Povely, příkazy a funkce Mikro BASIC

Povely	Příkazy			Funkce	Pseudoproměnné	Pomocná slova
LIST	LET	DISPL	WORD (expr.)	RND (expr.)	LEN	TO
LLIST	FOR NEXT	IF	MASK expr.	ABS (expr.)	TOP	STEP
RUN	GOTO expr.	END	WAIT expr.	HEX (hex. hodn.)	SIZE	
NEW	GOSUB expr.	STOP	BEEP expr.	INCHAR		
MONITOR	RETURN	CLEAR	O$ expr.	PEEK (expr.)		
RAM exp.	REM	CLS	I$ expr.	IN (expr.)		
LOAD	INPUT	CALL expr.		INM (expr.)		
SAVE	PRINT	POKE expr., expr.				
	LPRINT	OUT expr., expr.				
	TAB (expr.)	OUTCHAR expr.				
	HARD	BYTE (expr.)				

expr. = expression (výraz)

Dům techniky ČSVTS Ostrava uspořádá v letošním roce kursy pro uživatele systému SAPI 1 (mikropočítač JPR-1). V rámci kursu budou účastníkům dodány základní desky systému (centrální deska JPR-1). Informace podá
Dům techniky ČSVTS, nám. K. Marxe 5, 728 09 Ostrava